Universidad de las Ciencias Informáticas

Facultad 1

Desarrollo y mantenimiento de ROM Android para dispositivos móviles ensamblados en Cuba

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Autores: Natalí Martínez Sarduy

David Alejandro Reyes Milian

Tutores: Ing. Abel Alfonso Fírvida Donéstevez

Mtr. Allan Pierra Fuentes

Ciudad de La Habana, junio de 2015
"Año 57 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

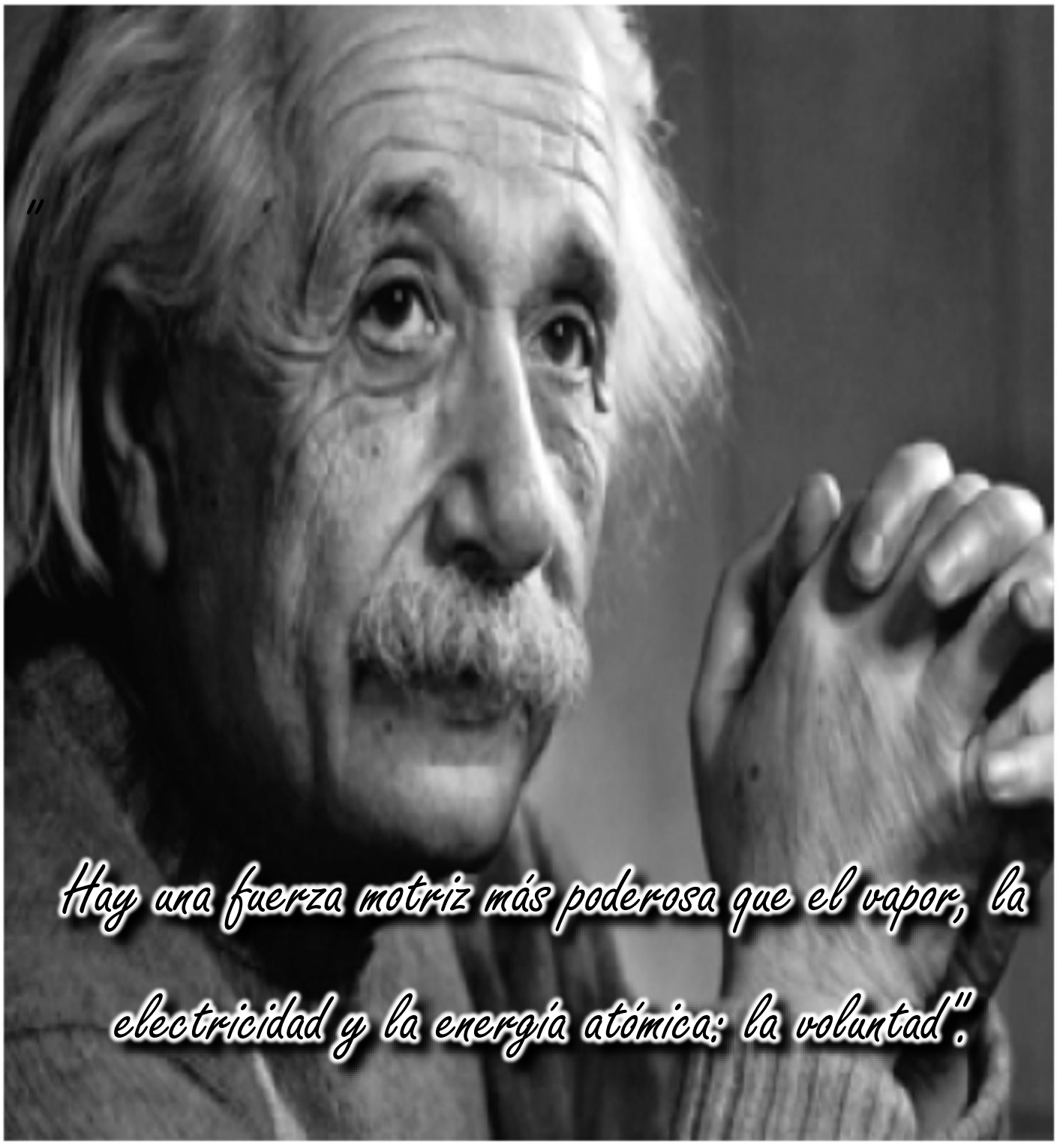
Para que así conste firmamos la presente a los _____ días del mes de _____ del año 2015.

Autores:

_____	_____
Natalí Martínez Sarduy	David Alejandro Reyes Milian

Tutores:

_____	_____
Ing. Abel A. Fírvida Donéstevez	Mtr. Allan Pierra Fuentes


"
Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad".

DEDICATORIA

A mi familia, principalmente a mis padres Joel, Bárbara y mi hermano Gabriel.

David Alejandro Reyes Milian

A Mauricio, mi niño lindo, por su cariño incondicional.

A mi mamá por ser mi apoyo, mi amiga y confidente.

A mi papá, por hacerme crecer y fomentar mis metas.

A mis abuelos del alma…

Natalí Martínez Sarduy

AGRADECIMIENTOS

A mis padres Bárbara y Joel por ser mi sustento y orgullo de toda la vida pasada y por venir.

A mi hermano Gabriel, por alimentar mis sueños y hacer de mi camino futuro algo más sencillo con su ejemplo y experiencias.

A mi china bella Natalí por ser la persona más maravillosa y perfecta que haya conocido jamás, por su sonrisa tierna, por ser la luz de mi vida y mi compañera en todos los sentidos.

A Miladis y toda la familia de Naty por su apoyo.

A Rydel, Raydel, Emmanuel, todos mis hermanos y compañeros de la vocacional, compañeros de clase, y a todas mis amistades que han sido imprescindibles en mi formación como persona.

A los profesores que me ayudaron y depositaron en mí pedazos de esperanza y conocimiento.

A mis tutores y al personal del centro CESOL por su apoyo.

A Héctor por su ayuda y dedicación durante el transcurso de este proyecto.

A la Universidad en general por permitirme cinco años intensos y de gran valor para mi vida profesional.

David Alejandro Reyes Milian

A mis padres por todo el esfuerzo realizado para alcanzar este sueño, a mi familia en general por su apoyo y cariño.

A mi David, por ser la persona extraordinaria que es… por llenar mi vida de amor e ilusiones, por ser mi soporte, mi guía, mi ejemplo…

A Bárbara, Joel y Gabriel por su cariño y por hacerme parte de la familia.

A mis amigas y amigos… de la infancia Yoda, Dana, Liane… a los que conocí en la UCI Clau, Mariam, Tania, Daneysy, Basi, Yani… por su paciencia, comprensión y las tardes de café…

A mis tutores Abel y Allan por su ayuda, a Héctor por su preocupación constante.

A Cesar González por ayudarnos con los dispositivos en la tesis.

A la UCI por brindarme la oportunidad de crecer como persona y profesional y a todos los profesores que lo hicieron posible.

Natalí Martínez Sarduy

RESUMEN

El presente trabajo muestra el proceso de desarrollo y mantenimiento de *ROMs* Android, haciendo énfasis en las dos vertientes principales para lograrlo: Crear *ROM* a partir del código fuente de Android y Crear *ROM* mediante el uso de herramientas conocidas como “Cocinas”.

Se realizó el estudio del estado del arte correspondiente a las herramientas, técnicas, métodos y recursos existentes en la creación de personalizaciones, así como la estructura interna de Android creando una caracterización sobre la que se basa el proceso propuesto.

Finalmente se crean *ROMs* para los dispositivos de prueba Nut Neko – SoC MT6589, Alcatel One Touch – SoC MTK6577 y un dispositivo virtual de Android como parte de la validación del proceso descrito.

Palabras claves:

Android, Código Abierto, *ROM*, Compilar, Cocinar.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Android es un conjunto de código abierto basado en el núcleo de Linux que abarca una amplia gama de dispositivos móviles y sus proyectos de desarrollo correspondientes liderados por Google (Android, 2013). Inicialmente desarrollado por Android Inc., que Google respaldó económicamente y más tarde compró en 2005. Android se dio a conocer en 2007, junto con la fundación Open Handset Alliance: un consorcio de empresas de tecnología como Google, fabricantes de dispositivos como HTC, Sony y Samsung, proveedores de servicios inalámbricos como Sprint Nextel y T-Mobile, y los fabricantes de *chipsets* como Qualcom y Texas Instruments, con el objetivo de desarrollar estándares abiertos para dispositivos móviles. (*Open Handset Alliance*, 2007)

Según el último informe publicado por la firma IDC los terminales con iOS y Android siguen dominando el mercado de los teléfonos inteligentes con un margen muy amplio; juntos representaban el 96,3% de la cuota de mercado de dispositivos móviles inteligentes a nivel mundial durante el cuarto trimestre de 2014. Por separado, en el presente trimestre iOS representaba el 14,8% de los dispositivos, mientras que Android el 81,5%. (Alvarado, 2015)

A pesar de la brecha tecnológica existente, Cuba se encuentra inmersa en un proceso de informatización que desde el año 2004 protagoniza el software libre como la única opción factible para aumentar los niveles de socio-adaptabilidad, seguridad, sostenibilidad y soberanía tecnológica (Fuentes, 2011). Es por esto que se apuesta por Android como tecnología para los dispositivos móviles, sin embargo es desarrollada por Google (empresa multinacional estadounidense) y ampliamente utilizada a nivel mundial por lo que cabe analizar las ventajas y riesgos de la adopción de esta tecnología en el escenario actual.

El país asimila actualmente una penetración constante de dicha tecnología, siendo usada en muchas esferas de la sociedad, incluso por usuarios con altas responsabilidades que manejan información sensible.

Los dispositivos móviles son distribuidos con una *ROM* (*Read Only Memory*) previamente instalada por el fabricante, esto significa que el usuario depende totalmente de las modificaciones o actualizaciones proporcionadas por el mismo. Las implementaciones de estas *ROMs* poseen particularidades de acuerdo a las necesidades y objetivos del proveedor. Por lo cual no existe certeza o conocimiento de cómo ocurren los procesos de comunicación entre el dispositivo y el resto de la red y servicios. Según la revista especializada

Ticbeat (Ticbeat, 2012), el 26% de aplicaciones móviles del Google Play acceden a información personal de usuarios y un 25% de las aplicaciones son identificadas como sospechosas. Esto representa un alto riesgo para la seguridad de la información sensible del usuario, pues no se tiene control sobre los servicios y aplicaciones instaladas en el dispositivo.

Por otra parte, los terminales son distribuidos con un conjunto de aplicaciones instaladas, que por lo general obedecen a los intereses propios de cada distribuidor. Dichas aplicaciones, afectan considerablemente el rendimiento del dispositivo, y pierden efectividad en el momento en que no son útiles para el usuario, tal es el caso de los servicios de Google (Google Maps, Google Alerts, Google Keep, Google Calendar y Google News), que actualmente no pueden ser usados en Cuba.

Algunas de estas aplicaciones no pueden ser desinstaladas por un usuario común, por lo que los problemas de rendimiento son inevitables sobre todo para dispositivos de gama baja.

Actualmente el movimiento de software libre, específicamente de desarrollo de aplicaciones para Android en el país, se encuentra en una etapa de proliferación, denotada por la aparición de un conjunto de aplicaciones alternativas que obedecen a las necesidades particulares de los usuarios cubanos. Tal es el caso del FNM (Formulario Nacional de Medicamentos de Cuba), la versión de la Ecured para móviles y Andariego, aplicación creada por GeoSi, una entidad autofinanciada del Grupo Empresarial GEOCUBA. Sin embargo, el usuario estándar no tiene el acceso ni conocimiento necesarios para instalar algunas de estas aplicaciones, por lo que se hacen inaccesibles para el mismo.

Los fabricantes o proveedores de las *ROMs* brindan generalmente actualizaciones *OTA* (*Over The Air*) periódicas para eliminar fallas de seguridad además de actualizaciones críticas al núcleo, mejoras importantes al funcionamiento y rendimiento del dispositivo, así como el uso de energía, recurso crítico en dispositivos móviles. Dichas actualizaciones no son accesibles para el país actualmente.

Por lo anteriormente planteado se concluye que existe la necesidad de estudiar y entender el proceso de creación de una personalización de Android para lograr mitigar las dificultades expuestas inherentes al contexto cubano.

Partiendo de esta problemática se plantea como **problema científico** ¿Cómo elevar los niveles de sostenibilidad, seguridad, socio-adaptabilidad y soberanía tecnológica en el proceso de adopción de las tecnologías móviles en Cuba?

Definiéndose como **objeto de estudio** el proceso de desarrollo de personalizaciones para Android, ubicando como **campo de acción** la creación de personalizaciones para Android.

Definiéndose como **objetivo general** desarrollar una distribución de software basado en Android, mediante un proceso de personalización de *ROMs* para aumentar los niveles de sostenibilidad, seguridad, socio-adaptabilidad y soberanía tecnológica.

Para dar cumplimiento al objetivo planteado se han definido los siguientes **objetivos específicos**:

- Analizar el marco teórico de la investigación relacionado con la creación de personalizaciones para Android.
- Diseñar el procedimiento de creación de una personalización de Android.
- Construir una personalización de Android.
- Probar la personalización de Android en los dispositivos de prueba Nut Neko, Alcatel One Touch y en *AVD* (*Android Virtual Device* o Dispositivo Virtual Android).

Se plantea como **idea a defender** que la creación de una personalización cubana para Android, incrementará los niveles de seguridad, socio-adaptabilidad, sostenibilidad y soberanía tecnológica sobre los dispositivos móviles ensamblados en Cuba.

Para dar cumplimiento a los objetivos específicos planteados se realizan las siguientes **tareas de investigación**:

- Estudio de la arquitectura de Android.
- Análisis y descripción de las particiones que conforman una *ROM* de Android.
- Estudio de las características de las *ROMs* existentes.
- Revisión de bibliografía asociada a la implementación de personalizaciones de Android.
- Estudio de las herramientas de desarrollo de personalizaciones de Android.
- Definición de las herramientas a utilizar para la implementación de la personalización.
- Diseño y elaboración del proceso de creación de una personalización de Android.
- Análisis de las necesidades actuales del contexto para determinar el conjunto de aplicaciones a portar en la *ROM*.

Diseño metodológico de la investigación:

Métodos Teóricos:

- Analítico – Sintético: Para analizar Android, su estructura, particiones y las aplicaciones que lo componen.
- Inductivo – Deductivo: Para determinar cuáles son los aspectos necesarios en la creación de una *ROM* a la medida para teléfonos ensamblados en Cuba.

Métodos Empíricos:

- Observación: Para la obtención de conocimiento e información acerca del comportamiento de las diferentes personalizaciones de Android.
- Encuesta: Para dar participación a la comunidad en la selección de las aplicaciones con las que contará la *ROM*, además de funcionalidades y servicios inherentes al contexto.

La estructura del presente trabajo de diploma consta de tres capítulos, donde se expone lo referente a la creación de personalizaciones para Android, el proceso de creación de la *ROM* y la implementación y prueba sobre dos dispositivos reales y uno virtual que demuestran la validez de la propuesta. Para cada capítulo se realizan la introducción y las conclusiones parciales.

En el primer capítulo se ofrece un estudio del estado del arte, donde se exponen algunos conceptos relacionados con la tecnología Android, el desarrollo de personalizaciones y la estructura de las mismas.

En el segundo capítulo se expone detalladamente la propuesta del proceso de desarrollo de la *ROM* para Android.

En el tercer capítulo se brinda la solución implementada para tres casos de estudio prácticos sobre escenarios reales partiendo de los dispositivos de prueba, que evalúan y validan la propuesta.

CAPÍTULO 1. **FUNDAMENTACIÓN TEÓRICA**

En este capítulo se definen los elementos fundamentales necesarios para el entendimiento del proceso de desarrollo y mantenimiento de una *ROM*. Se realiza además el estudio del estado del arte de las personalizaciones Android, profundizando en las características, ventajas y desventajas de cada una. Se realiza además un estudio de las herramientas y tecnologías necesarias para la implementación de una personalización, así como el proceso de compilación del código fuente de Android.

1.1. Conceptos Fundamentales

A continuación se definen los conceptos fundamentales que constituyen la base sobre la que se definirán los procesos y las técnicas de creación de *ROMs* así como todos los aspectos relacionados con el tema.

ADB

Entre las herramientas más conocidas e importantes para interactuar con los dispositivos móviles se encuentra *ADB* (*Android Debug Bridge*), que permite administrar emuladores o dispositivos Android además de ser utilizada para depurar aplicaciones en desarrollo u otras opciones avanzadas. Permite obtener una consola de usuario *root* dentro del dispositivo, abriendo las puertas a muchas operaciones necesarias para los desarrolladores. Tiene una estructura cliente-servidor que está formada por tres componentes:

- Un cliente, que se ejecuta en el equipo de desarrollo.
- Un servidor, que se ejecuta como un proceso en segundo plano en su equipo de desarrollo.
- Un *daemon*[1], que se ejecuta como un proceso en segundo plano en cada emulador o dispositivo.

Para utilizar *ADB* solo es necesario disponer del ejecutable generalmente encontrado bajo la carpeta */platform-tools/* dentro del *SDK* de Android[2].(Android Developers, 2015)

[1] Daemon: Un demonio (nomenclatura usada en sistemas UNIX y *UNIX-like*), servicio (nomenclatura usada en Windows) o programa residente (nomenclatura usada en MS-DOS) es un tipo especial de proceso informático no interactivo, es decir, que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario**.**

[2]SDK de Android: Kit de desarrollo de software que permite a los desarrolladores crear aplicaciones para la plataforma Android.

Los procesos de instalación de *ROMs* sobre un terminal, siempre poseen un factor de riesgo y se debe tener mucho cuidado con el nivel de complejidad y el orden de las modificaciones que se realizan, pues el dispositivo puede terminar *brickeado*.

Brick

Brick, ladrillo en inglés, en el ámbito de los dispositivos móviles, se refiere al estado en que un dispositivo deja de funcionar, perdiendo incluso los protocolos o formas de comunicación con el ordenador. Cuando un dispositivo se encuentra en este estado se dice que está *brickeado*. Dicha situación tiende a ser compleja ya que se imposibilita la interacción con el dispositivo y su reparación puede necesitar el uso del hardware específico con estos fines. En ocasiones el daño resulta irreparable para el dispositivo, quedando inutilizable, de aquí la metáfora de ladrillo.

1.2. Caracterizando Android

Se hace necesario un estudio de la estructura interna de Android para asociar las herramientas y funcionamiento en aras de esclarecer el ciclo de desarrollo de una *ROM*.

1.2.1. Arquitectura de Android

Android está formado por varias capas que facilitan al desarrollador la creación de aplicaciones, permitiendo acceder a las capas más bajas de la jerarquía mediante el uso de librerías o *APIs*[3] de modo que se logre la abstracción necesaria evitando así programar a bajo nivel las funcionalidades específicas del hardware del dispositivo. Cada capa utiliza elementos de la capa inferior para realizar sus funciones, es por ello que a este tipo de arquitectura se le conoce también como pila. La arquitectura de Android contiene cinco capas fundamentales.

A continuación se muestra el diagrama de la arquitectura de Android tomada del sitio oficial de AOSP.

[3]Una *API* (*Application Programming Interface*) es un conjunto de rutinas, protocolos y herramientas para construir aplicaciones informáticas. Expresa un componente de software en término de sus operaciones, entradas, salidas y tipos de datos.

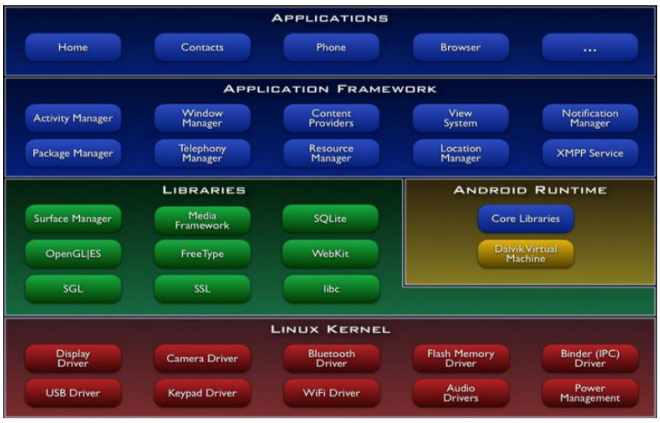


Imagen 1 Capas de la arquitectura de Android.

Núcleo de Linux (*Linux Kernel*)

El núcleo o *kernel* de Android está formado por el núcleo de Linux versión 2.6. Esta capa actúa como base del sistema operativo proporcionando servicios como la gestión de seguridad, el manejo de la memoria, el multiprocesamiento y el soporte de *drivers* para dispositivos. Esta capa del modelo actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del mismo.

Android Runtime

Capa basada en el concepto de máquina virtual utilizado en Java. Dadas las capacidades limitadas (poca memoria y procesador limitado) no fue posible utilizar una máquina virtual Java estándar. Google tomó la decisión de crear una nueva, la máquina virtual *Dalvik*, que respondiera mejor a estas dificultades. A continuación se muestran algunas características de la máquina virtual *Dalvik* que facilitan esta optimización de recursos:

- Ejecuta ficheros *Dalvik* ejecutables (.*dex*) –formato optimizado para ahorrar memoria.
- Está basada en registros.
- Cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual *Dalvik*.
- Delega al núcleo de Linux algunas funciones como *threading*[4] y el manejo de la memoria a bajo nivel.

[4] *Threading*: Función de algunos sistemas operativos que permite el procesamiento concurrente a partir del uso de hilos del núcleo.

A partir de Android 5.0 se reemplaza *Dalvik* por ART. Esta nueva máquina virtual consigue reducir el tiempo de ejecución del código Java hasta en un 33%.

Librerías Nativas (*Libraries*)

Contiene las librerías utilizadas por Android. Estas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes ubicadas en este nivel, se encuentran:

- **Librería libc**: Incluye todas las cabeceras y funciones según el estándar del lenguaje C.
- ***Surface Manager***: Es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- ***OpenGL/SL* y *SGL***: Representan las librerías gráficas; por tanto, sustentan la capacidad gráfica de Android. *OpenGL/SL* maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, *SGL* proporciona gráficos en 2D, por lo que será la librería más utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.
- ***Media Libraries***: Proporciona todos los *códecs* necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)
- ***FreeType***: Permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
- **Librería *SSL***: Posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
- **Librería *SQLite***: Creación y gestión de bases de datos relacionales.
- **Librería *WebKit***: Proporciona un motor para las aplicaciones de tipo navegador y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Marco de trabajo de Android (*Aplication Framework*)

Contiene fundamentalmente el conjunto de herramientas comunes para el desarrollo de aplicaciones. Toda aplicación desarrollada para Android utiliza el mismo conjunto de *APIs* y el mismo *framework*. Entre las *APIs* más importantes ubicadas en este nivel, se encuentran:

- ***Activity Manager*:** Gestionan el ciclo de vida de las aplicaciones en Android.

- ***Window Manager*:** Gestiona las ventanas de las aplicaciones y utiliza la librería *Surface Manager*.
- ***Telephone Manager*:** Incluye todas las *APIs* vinculadas a las funcionalidades propias de telefonía (llamadas, mensajes, etc.).
- ***Content Provider*:** Permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible por otras aplicaciones.
- ***View System*:** Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, *check-boxes*, tamaño de ventanas, control de las interfaces mediante teclado, etc. Incluye también algunas vistas para las funcionalidades más frecuentes.
- ***Location Manager***: Posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- ***Notification Manager***: Mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión inalámbrica disponible, ubicación en un punto determinado, etc.
- ***XMPP Service***: Colección de *APIs* para utilizar este protocolo de intercambio de mensajes basado en XML.

Aplicaciones (*Aplications*)

En esta capa se encuentran las aplicaciones accesibles por el usuario, es la capa más superficial de la arquitectura, por lo que es la menos protegida. Contiene un conjunto mínimo de aplicaciones base como son un cliente de correo, calendario, programa de *SMS*, mapas, navegador, contactos, además de todas las aplicaciones impuestas por el proveedor de la *ROM*. Dichas aplicaciones son escritas generalmente con el lenguaje Java.

1.2.2. Particiones en Android

La memoria física de un dispositivo móvil o tableta, al igual que el disco duro de un ordenador, puede dividirse en varias fracciones lógicas, conocidas como particiones. En Android cada partición tiene un propósito específico que permite hacer divisiones lógicas en el sistema, logrando aumentar la seguridad y estabilidad. La mayoría de estas particiones se crean en la memoria interna del dispositivo, una memoria de estado sólido (*flash*), conocida como *NAND*. Si un dispositivo tiene tarjeta *SD* para extender su espacio de almacenamiento,

la tarjeta en cuestión representará otra partición. A continuación se muestra información relacionada con las particiones por defecto de Android.

/boot

Gestiona el arranque del dispositivo, contiene el *bootloader* y el núcleo. Sin esta partición, el dispositivo no sería capaz de iniciar. Es una partición crítica, que debe tratarse con cuidado. Generalmente es la partición más delicada, debido a que en caso de ser dañada se puede perder la comunicación con el dispositivo entrando en el estado denominado *brick* o de ladrillo.

/system

Contiene programas y configuraciones que el fabricante u operador móvil suministra inicialmente con el teléfono, siendo su contenido montado por el sistema en modo de lectura exclusiva, lo que indica que el usuario puede leer los archivos que contiene pero no puede modificarlos, con independencia de los permisos asignados a los mismos.

/recovery

Puede considerarse como una partición alternativa a la de inicio (***/boot***) que permite iniciar el dispositivo en un modo especial llamado **modo de recuperación** (*recovery mode*). Esta partición aloja una aplicación del mismo nombre que permite realizar tareas de mantenimiento, recuperación de datos, instalación de *ROMs* entre otras operaciones avanzadas en dependencia de la aplicación de recuperación que se disponga. Por lo general la aplicación instalada en la partición de recuperación es la llamada *stock recovery* pero puede ser sustituida por otras herramientas avanzadas como las conocidas ClockworkMod (CWM) y TWRP, que serán abordadas más adelante en este documento.

/data

Almacena los archivos asociados a cada una de las aplicaciones instaladas, encargándose Android de crear un directorio para cada una de ellas, a las que asigna permisos de acceso de forma que sólo la aplicación a la que pertenecen pueda acceder a estos.

Partición del núcleo

Montada habitualmente a partir de la carpeta */sys*, contiene el núcleo del sistema, así como los módulos, librerías asociados a éste y los *drivers* o archivos de cada uno de los dispositivos, tales como la propia CPU

(se encuentran los archivos relacionados con las velocidades máxima y mínima de la CPU en la carpeta */sys/devices/system/cpu*, por ejemplo).

/cache

Partición donde Android guarda los datos a los que el usuario accede con frecuencia para aumentar el rendimiento. Esto hace que tareas frecuentes funcionen mucho más rápido que otras que no sean tan habituales.

/misc

Contiene información adicional relacionada con la configuración de sistema, en forma de "interruptores" de encendido/apagado. Esta información puede incluir el *CID* (*Carrier or Region ID* o identificador del operador o región), la configuración USB o ciertos ajustes hardware. Es una partición importante que en caso de pérdida o corrupción puede provocar que algunas características del dispositivo dejen de funcionar.

/sdcard

Pertenece a la tarjeta *SD*. Es donde se guardan los datos que se quieran almacenar, como archivos multimedia, documentos, etc. Además muchas aplicaciones instaladas por el usuario guardan aquí todos los datos y configuraciones.

/sd-ext

Es una partición adicional que actúa como una extensión de la partición ***/data***, cuando se utilizan aplicaciones como APP2SD+ o data2ext. Es especialmente útil en dispositivos con una memoria interna muy pequeña. Puede usarse para instalar aplicaciones más allá de las que la memoria interna permite, siempre y cuando la *ROM* que se tenga instalada, tenga activada esta capacidad.

1.3. Antecedentes de la creación de *ROM* Android

Primeramente es necesario definir el significado de *ROM* en el ámbito de Android.

ROM (Read-Only Memory)

Android se distribuye en paquetes de código haciendo posible una expansión compacta con seguridad ante fallos gracias a las sumas de verificación que contienen, llamados *ROMs*. En informática, *ROM* se denomina a la memoria de sólo lectura donde se instala el *firmware* que permite el funcionamiento de un dispositivo

cualquiera. Es una zona sensible en la que se alberga el sistema operativo, lugar en que cada fabricante suele incluir adaptaciones y configuraciones adecuadas para lograr la mejor integración posible con el hardware en cada terminal. En el ámbito de Android suele utilizarse este término para referirse al **archivo que contiene el sistema operativo** listo para ser transferido a la memoria *flash* del teléfono, proceso llamado comúnmente *flashear*[5].

La creación de una *ROM* Android para dispositivos móviles es el proceso mediante el cual se obtiene un **paquete instalable** (*flasheable*) de **Android**, compatible con el dispositivo para el cual se crea. Dicho proceso, así como la *ROM* obtenida, son generalmente dependientes del hardware específico del dispositivo anfitrión, por lo que el proceso de creación de *ROM* toma como una de sus entradas principales el modelo y especificaciones del dispositivo destino. Existen dos métodos o formas fundamentales para la obtención de una *ROM*:

- Creación a partir de una *ROM* existente (llamado cocinar *ROM*).
- Compilación del código fuente de Android para el dispositivo deseado.

Ambos métodos poseen características de interés para la creación de una *ROM* por lo que serán abordados a continuación.

1.3.1. Creación de ROM a partir de ROM existente

Se denomina “cocinar una *ROM*” al proceso mediante el cual se crea una *ROM* Android a partir de una *ROM* previa con el uso de herramientas conocidas como “cocinas”. Dicho proceso, requiere de un paso esencial que es la selección de la *ROM* que se desea portar al dispositivo destino.

Existen varios tipos fundamentales de *ROMs* (Androizados, 2014) sobre las cuales se centra el proceso de selección:

OEM, *Original Equipment Manufacturer* (Fabricante Original del Equipo). Creadas específicamente por el fabricante original del dispositivo por lo que suelen ser más estables y libres de errores. Generalmente son re-ediciones de *ROMs* para dispositivos específicos, aunque solo pueden ser instaladas en éstos. Son también llamadas *Stock ROMs.*

[5] Flashear: Instalar o actualizar el sistema de arranque del dispositivo.

AOSP, *Android Open Source Project* (Proyecto de Código Abierto de Android). Creadas por los desarrolladores a partir del código fuente de Android. Son populares porque representan la forma de soporte, en ocasiones la única, de los dispositivos obsoletos, además, la diversidad de dispositivos existentes hace difícil, si no imposible, que las liberaciones de actualizaciones Android ocurran simultáneamente para todos los dispositivos, sin embargo, a través de una *ROM* AOSP es posible actualizar el SO. Pueden ser menos estables por incompatibilidades de hardware. Dicho tipo de *ROM* es conocido también como *Custom ROM* (*ROM* personalizada o personalización de Android). Estas personalizaciones aportan algunas mejoras en cuanto a rendimiento y herramientas avanzadas principalmente para desarrolladores.

ROMs existentes

Existe una amplia gama de *ROMs* personalizadas de Android, cada una buscando satisfacer las necesidades de los clientes específicos del fabricante o de usuarios en general. Según (Martínez, 2012), las personalizaciones más importantes son:

CyanogenMod

Personalización que toma como directrices favorecer el rendimiento y portar características nuevas, así como versiones de Android a dispositivos, generalmente antes de la liberación oficial por parte de los fabricantes. La interfaz visual es parecida a la versión sin modificaciones de Android que puede observarse en cualquier Nexus, no obstante, posee funciones añadidas, que la convierten en algo diferente y único. (CyanogenMod Wiki, 2014)

CyanogenMod se desarrolla principalmente en cinco versiones paralelas y activas:

- CyanogenMod 10.0 (Android 4.1)
- CyanogenMod 10.1 (Android 4.2)
- CyanogenMod 10.2 (Android 4.3)
- CyanogenMod 11.0 (Android 4.4)
- CyanogenMod 12.0 (Android 5.0.2)

Las variantes del *firmware* se dividen en categorías:

- Estable: Es la variante de eficacia probada del *firmware* demostrando ser en su mayoría libre de errores y adecuada para el uso diario.

- Candidato a Estable: Un *Release Candidate* (RC) puede no ser la versión final, pero sí un candidato que no tiene defectos o errores fatales.
- *Nightlies*: Estas *ROMs* son generalmente de prueba, no destinadas al uso del usuario medio. Tienen como propósito validar funcionalidades no probadas que según su complejidad pueden afectar el dispositivo de forma permanente. No está recomendado su uso.

AOKP

ROM basada en CyanogenMod de interfaz parecida pero incluye más funciones avanzadas y de personalización (Smart Mobile Phone Solutions, 2012).

Ventajas:

- Ofrece gran cantidad de opciones de configuración sin necesidad de instalar más aplicaciones como BusyBox que requieren de un usuario con privilegios *root.*
- Publica actualizaciones regulares de versiones oficiales estables.
- Permite configurar parámetros avanzados de la CPU a partir de un nuevo núcleo (*Faux*), que permite interactuar con la velocidad máxima y mínima del CPU así como la gestión del escalado y el voltaje.

Desventajas:

- Soporte sólo para terminales de gama alta.
- Requiere de conocimientos medio-avanzados para instalarla.

MIUI

Desarrollada por Xiaomi Tech, MIUI ofrece una interfaz diferente y renovada (algo parecida a iOS), siendo esta su principal propuesta. Es la *ROM* que utiliza Xiaomi en sus teléfonos inteligentes (Smart Mobile Phone Solutions, 2012).

Ventajas:

- Aplicaciones funcionales propias de MIUI.
- Número ilimitado de escritorios.
- Actualizaciones continuas.

Desventajas:

- Interfaz similar a la de iOS.

Paranoid Android:

Una de las *ROMs* Android más populares entre los clientes, su AOSP fue diseñado inicialmente de un modo similar a CyanogenMod, pero ha cambiado con el paso a Jelly Bean 4.2.1. Hoy está disponible en dispositivos Nexus y OPPO. Su última versión es Android 4.4.2 KitKat Paranoid Android Custom *ROM*.

Ventajas:

- Opciones de personalización con *Hybrid Engine*[6], *Pie Control*[7] y *HALO*[8].
- Facilidad de instalación.

Desventajas:

- Ausencia de versiones estables para la mayoría de los dispositivos.
- Pocas opciones de personalización avanzada en comparación con otras *ROMs*.

Herramientas para cocinar *ROMs*

Una cocina de *ROMs* es un conjunto de herramientas y *scripts* destinados a la modificación de *ROMs* que permiten la modificación a partir de procedimientos generalmente *ad-hocs*[9] y de ingeniería inversa. Normalmente proveen al usuario de *shortcuts* (atajos) para lograr operaciones avanzadas con la *ROM* sin necesidad de conocimientos profundos en el tema. Existen disímiles herramientas para cocinar *ROMs* que se presentan tanto *online* como *offline*.

Cocinas *online*

Estas herramientas son trabajadas de manera *online*, por lo que no requieren de recursos locales, entornos de trabajo pre configurados o conocimientos extra para lograr cocinar una *ROM*, convirtiéndose en una opción factible, sin embargo al estar sujetas a las condiciones del entorno de trabajo del servidor y los recursos de que disponga el mismo, pueden no ser adecuadas para determinadas tareas.

[6]Hybrid Engine: Permite elegir la interfaz de usuario de cualquier aplicación.
[7] Pie Control: Herramienta utilizada para ocultar los botones de navegación cuando no están en uso.
[8]HALO: Proporciona notificaciones siempre visibles que permiten interactuar con las ventanas pop-up desde cualquier pantalla.
[9] Ad-hoc: Generalmente se refiere a una solución específicamente elaborada para un problema o fin preciso.

Existen varias cocinas *online* que aunque poseen particularidades y diferencias, comparten el procedimiento general:

1. Se establece la *ROM* que se desea modificar.
2. Se realizan las modificaciones deseadas sobre esta.
3. Una vez realizadas todas las acciones, la aplicación vuelve a comprimir la *ROM* y la ofrece para su descarga.

Entre las posibilidades que ofrece una cocina *online* se encuentran añadir o eliminar aplicaciones del sistema instaladas en una *ROM*, elegir un *bootsplash* (imagen de arranque) distinto, la activación o desactivación del sonido en el arranque, agregar pequeñas utilidades desarrolladas por la comunidad para mejorar la capacidad, modificar los iconos, temas, la barra de estado y notificaciones (fuentes, transparencia, etc.), así como modificar las formas de *wake-up*[10]del terminal. El uso de las cocinas *online* ofrece las siguientes ventajas:

- No es necesario que el usuario esté familiarizado con el amplio conocimiento relacionado con el proceso para cocinar la *ROM*.
- No es necesario preparar un entorno de trabajo local, ni poseer medios o recursos para llevar a cabo el proceso.

Como desventajas pueden observarse:

- La cantidad de terminales soportados por lo general es baja.
- Los sitios web pueden y suelen ser inestables.
- Las posibilidades están limitadas a las opciones publicadas en el sitio y por lo general son reducidas.

Cocinas online:

- MoDaCo Kitchens
- Cook Android
- Cocina Online Android
- Feeder.pk Nexus One Kitchen

[10]Wake-up (despertar en español) se refiere a despertar el dispositivo del estado de inactividad.

- Multikitchen Online
- Cook Android
- AndroidRTK
- RomKitchen

Cocinas offline

Por otra parte existen herramientas de uso local (*offline*), que si bien requieren de recursos propios y más conocimiento, proporcionan más opciones y el procedimiento es más limpio, en ocasiones *debugueable* y visible. Entre las múltiples opciones que permiten estas herramientas se listan algunas a continuación:

- *Rootear* el dispositivo o desbloquear el usuario administrador del sistema.
- Añadir soporte para BusyBox[11].
- Deshabilitar sonidos en las pantallas de *booteo*.
- Añadir capacidad para *wireless tethering*[12].
- Firmar *apks*[13].
- Añadir herramientas y opciones avanzadas a la *ROM* como *Apps2SD*, *Bash*, *Nano*, etc.

Una de las herramientas más populares en la comunidad es la llamada ***dsixda's Android Kitchen*** cuyo código abierto representa una verdadera biblia dentro del tema de las cocinas de *ROMs*, de gran utilidad sobre todo para estudiar el procedimiento y añadir funcionalidades a la misma. Está compuesta principalmente por *scripts* en *bash* por lo que resulta bastante sencillo realizar modificaciones a la cocina.

1.3.2. Compilación de código fuente Android

[11]BusyBox es un binario que encapsula múltiples funciones nativas de GNU/Linux proporcionando su uso al usuario dentro del dispositivo Android.

[12] Wireles *tethering*: Se refiere a la conexión de un dispositivo a otro. En el contexto de los dispositivos móviles, *tethering* permite compartir la conexión a Internet del teléfono o tableta con otros dispositivos como ordenadores portátiles.

[13] En Android es necesario que todas las aplicaciones que se instalen estén firmadas digitalmente con la clave privada del desarrollador de la aplicación, permitiendo identificar a los desarrolladores y que se puedan establecer relaciones seguras entre las aplicaciones.

Android es un software *Open Source* o de código fuente[14] abierto, esto significa que el código fuente está disponible y es modificable, tanto por usuarios comunes, como por empresas u otras entidades. Dicha característica lo convierte en un ecosistema donde los desarrolladores pueden implementar sus ideas innovadoras con mínimas restricciones, sin embargo las modificaciones sin control pueden llevar a implementaciones incompatibles. Para prevenir esto, AOSP mantiene paralelamente el *Android Compatibility Program* o Programa de Compatibilidad de Android (en adelante PCA) que dicta el significado de qué es una implementación Android-compatible y los requerimientos que tienen que ser cumplidos por los fabricantes de dispositivos para alcanzar dicho estatus. Cualquier usuario puede y podrá usar el código fuente de Android con cualquier propósito, sin embargo, para formar parte del ecosistema de aplicaciones que se está construyendo alrededor de Android los desarrolladores de dispositivos deben participar en el PCA.

El proceso de compilación de Android para un dispositivo requiere:

- Que el dispositivo sea Android-compatible según lo dictado por el PCA.
- Poseer los *drivers* del hardware específico del dispositivo, así como el núcleo compilado para el mismo.

Dichos *drivers* son únicos para cada pieza de hardware e independientes de Android, por lo que la falta de estos provocará incompatibilidad con el sistema operativo traducida en el mal funcionamiento del dispositivo. Son generalmente proporcionados por su fabricante, sin embargo en ocasiones no ocurre de esta manera y es necesario obtenerlos a partir de métodos no convencionales.

1.4. Modos de conexión en terminales Android

Para interactuar con dispositivos móviles es necesario tener interfaces de comunicación de bajo nivel de forma que se garantice la comunicación en meta-estados no inicializados. Con este fin existen varios protocolos de comunicación que varían entre fabricantes e implementaciones. A continuación se muestran algunos de estos protocolos.

Bootloader

[14] El código fuente de un programa informático (o software) es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está escrito por completo su funcionamiento.

El *bootloader* es un programa que carga el sistema operativo o entorno de ejecución de un dispositivo después de terminar las rutinas de autocomprobación. Inicia el dispositivo en un estado en que se elige en que partición comenzar la ejecución del núcleo. Esto permite seleccionar por ejemplo iniciar con otro núcleo de Linux o comenzar otro sistema operativo. Debido a la importancia del *bootloader* como componente esencial en el proceso de *booteo*, es guardado en memoria no volátil, tal como la memoria *flash*. Los *bootloaders* son escritos por los fabricantes de hardware y están especializados en el hardware que ejecutan. En dispositivos Android, el *bootloader* (llamado *Android bootloader*) típicamente inicia en modo recuperación, en modo *Bootloader* o inicia el sistema operativo normalmente. Frecuentemente posee un modo básico interactivo que puede ser disparado presionando “volumen abajo” mientras se ejecuta(XDA Developers, 2015).

Fastboot

Fastboot es una pequeña herramienta contenida en el SDK de Android que puede ser usada para instalar particiones en un dispositivo. Es una alternativa al modo de recuperación para realizar instalaciones y actualizaciones.

Debido a que *fastboot* puede iniciar en el dispositivo incluso antes que Android (y puede además ser ejecutado sin tener Android instalado), es útil para actualizar el *firmware* rápidamente, sin tener que usar el modo de recuperación. De hecho, es frecuentemente el modo preferido de instalación de imágenes de recuperación en muchos dispositivos. Puede además ser utilizado para operaciones de desarrollo como desbloquear el *bootloader* de los dispositivos Nexus de Google. No todos los dispositivos soportan *fastboot*, pero la mayoría lo hace.

Existen dos maneras de usar *fastboot*: desde la computadora o desde el dispositivo. Típicamente el dispositivo es primeramente conectado a la computadora utilizando un cable USB y luego se *bootea* el dispositivo en “modo *fastboot*”. En la computadora se ejecuta entonces la herramienta para ejecutar los diferentes comandos (CyanogenMod Wiki, 2014).

1.5. Herramientas de *flasheo* de *ROMs*

Luego de crear o modificar (cocinar) una *ROM* generalmente se procede a instalarla en un dispositivo a través del procedimiento denominado flasheo o instalación de *ROM*. Para ello existen herramientas dependientes del modelo y fabricante del dispositivo. Las herramientas con este fin en caso de existir son obtenidas de manos del propio fabricante del dispositivo.

Odin

Odin es un software de Samsung que se filtró hace algunos años y tiene la capacidad de instalar el *firmware* en los dispositivos Samsung. Es una herramienta muy poderosa que puede proporcionar mucho beneficio, pero a la vez es también muy peligrosa. En las manos equivocadas puede dañar el dispositivo instantáneamente. Es por eso que se necesita tener mucho cuidado cuando se usa y asegurarse de que se conoce lo que se está haciendo antes de usarla. *Odin* sin embargo, puede resultar la última esperanza para usuarios de Samsung que tengan un dispositivo inutilizado. *Odin* además permite instalar *ROMs* oficiales en el dispositivo, opción de gran valor, pues con frecuencia las actualizaciones de Android son liberadas primero en ciertos países y con esta herramienta podrían ser instaladas en el dispositivo.

MTK Droid Tools

Otra de las herramientas conocidas es MTK Droid Tools desarrollada para los dispositivos MT6592, MT6582, MT6589T, MT6577, MT6575, MT6572 y la mayoría los dispositivos MTK relativamente modernos. Permite crear copias de seguridad del *firmware* del dispositivo, además esta herramienta es capaz de crear y utilizar copias de seguridad de CWM (conocidos también como *NANDroid backups*[15]). También es posible determinar las especificaciones reales del dispositivo así como cambiar o restaurar el IMEI. La herramienta permite además crear un archivo *scatter*[16] para guardar la disposición exacta en memoria de las particiones en un dispositivo. Es muy importante tener disponible esta opción ya que muchos dispositivos MTK a pesar de poseer el mismo SoC[17] no mantienen la distribución lógica de sus particiones por lo que el archivo de dispersión cambia. La mejor opción es utilizar el archivo del propio dispositivo.

SP Flash Tool

Smart Phone Flash Tool es una herramienta desarrollada por MediaTek Inc. Permite principalmente tres funcionalidades:

---

[15] Un *NANDdroid Backup* (copia de seguridad *NANDdroid*) es una copia de seguridad del sistema Android en un dispositivo. NANDdroid es una estructura de directorios utilizada para crear copias de seguridad de sistemas Android. Es accesible a través del modo de recuperación permitiendo hacer una copia exacta del estado del sistema en el dispositivo (XDA Developers, 2015).
[16] *Scatter File*: Un archivo scatter o archivo de dispersión, en términos Android, es un fichero que describe la distribución de regiones en memoria en un dispositivo Android específico que ejecute la arquitectura *ARM* de MediaTek (Smart Mobile Phone Solutions, 2012).
[17] SoC (System on Chip): Circuito integrado (IC) que integra todos los componentes de un ordenador u otro sistema electrónico en un solo chip.

1. Instalar *ROM*: Instala *ROMs* personalizadas u oficiales en el terminal, de igual manera permite instalar versiones más o menos recientes de Android.
2. Formatear el dispositivo: Limpia completamente el contenido de cada una de las particiones definidas en el archivo de dispersión del terminal en cuestión. Sólo es necesario cargar el archivo de dispersión y ejecutar la opción *Format.*
3. Instalar sistema de recuperación: Instala solamente el sistema de recuperación avanzada, para esto de igual manera es necesario contar con el archivo de dispersión.

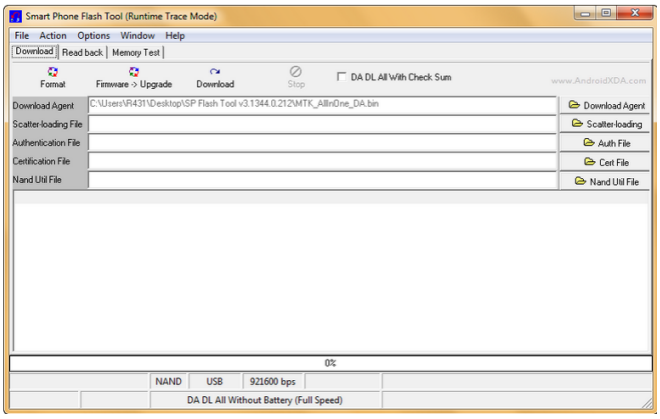


Imagen 2 SP Flash Tool

SP Flash Tool en sus versiones más recientes permite modificar cada una de las particiones descritas en el archivo de dispersión, ya sea para formatear, instalar u ambas opciones. Esta herramienta presupone la existencia del archivo de dispersión que se genera con la herramienta MTK Droid Tools.

1.6. Herramientas de recuperación para dispositivos Android

Todo dispositivo Android posee, en su estado original, una partición llamada *recovery* que permite iniciar el dispositivo en un modo especial con igual nombre. Esta herramienta permite realizar acciones sobre las particiones no montadas del dispositivo, es por ello que se necesita iniciar en este estado especial para tener acceso. Por defecto Android posee un sistema de recuperación, llamado *stock recovery* o sistema de recuperación de fábrica, sin embargo dicha herramienta no explota al máximo las opciones disponibles en este estado por lo que se han creado herramientas avanzadas. Las llamadas *custom recoverys* (sistema de recuperación modificado) son una ampliación en cuanto a funcionalidades y son muy importantes en el

proceso de instalación de *ROMs* Android. Dos de las herramientas más reconocidas son ClockworkMod (CWM) y TWRP.

ClockworkMod (CWM)

Creada por Koushik "Koush" Dutta, es una aplicación avanzada de recuperación que sustituye la aplicación de recuperación por defecto de Android en caso de ser instalada. Permite sobrescribir valores de recuperación del dispositivo, así como aplicar cambios de software. Permite crear y restaurar copias de seguridad o reemplazar totalmente el sistema operativo del dispositivo. Luego de instalado, el software es accesible *booteando* el dispositivo en modo *recovery*; esto se logra mediante combinaciones de teclas que cambian entre dispositivos o mediante herramientas como el ADB a través del siguiente comando.

```
adb reboot recovery
```

Una vez reiniciado, se obtendrá la pantalla inicial de la versión de CWM o TWRP instalada.

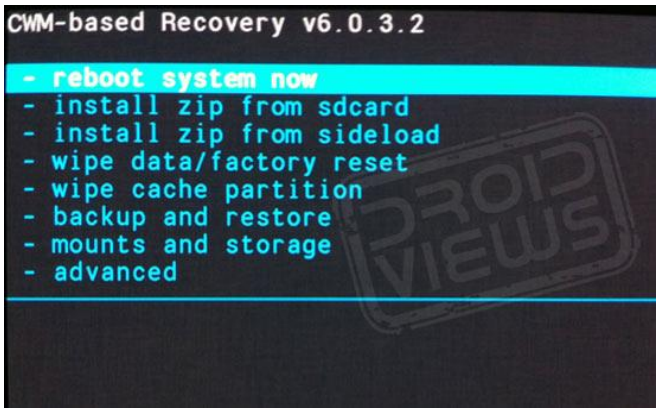


Imagen 3 Interfaz de usuario de CWM

Para una lista detallada de las funciones que permite la herramienta consulte el anexo #3.

TWRP (Team Win Recovery Project)

Es una herramienta de recuperación de código abierto para dispositivos basados en Android. Proporciona una interfaz táctil que permite a los usuarios instalar *firmwares* de terceros y realizar copias de seguridad del sistema actual. TWRP ofrece además la posibilidad de elegir qué particiones resguardar (a través de copias de seguridad) o restaurar (TeamWin, 2015).Puede consultar una lista de opciones disponibles en el documento anexo #4.

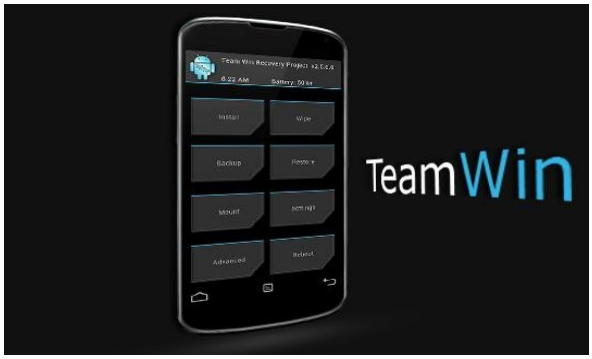


Imagen 4 Pantalla de inicio de TWRP

1.7. Modelado de procesos

Un proceso es una acción o sucesión de acciones continuas regulares, que ocurren o se llevan a cabo de una forma definida, y que llevan al cumplimiento de algún resultado; una operación continua o una serie de operaciones. En el ámbito de la investigación se puede considerar un proceso como un conjunto de actividades interrelacionadas, cuya salida final es la conformación de un bien o un servicio para un cliente que puede ser interno o externo de la organización.

Para el modelado del proceso de personalización de Android se utilizará la notación BPMN.

1.7.1. Notación BPMN

La notación BPMN del inglés *Business Process Modeling Notation* (Notación de Modelado de Procesos de Negocio) es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de actividades. Esta notación define un tipo de diagrama de procesos de negocio que se utiliza para modelar procesos de negocio (en inglés *Business Process Diagram* o BPD), así como los sub-procesos y tareas que lo componen.

Entre sus ventajas se encuentran:

- Considera un único diagrama para la representación de los procesos.
- Permite modelar los procesos de una manera unificada y estandarizada.
- Está diseñada para modelar procesos manuales, automáticos, físicos o virtuales.
- Crea un enlace entre la etapa de diseño e implementación.

Desventajas:

No incluye:

- Estructuras de organización y recursos.
- Reglas del negocio.
- Un mecanismo para el almacenamiento.

1.7.2. Herramienta de modelado de procesos

Visual Paradigm es una herramienta de ingeniería de software asistida por computadoras para apoyar al proceso de desarrollo de software. Soporta estándares de modelado tales como Lenguaje Unificado de Modelado (UML), Diagramas de Flujo de Datos (DFD), Diagramas de Procesos de Negocio utilizando la notación BPMN. Esta herramienta soporta y facilita el trabajo de los equipos de desarrollo durante la captura de requisitos, planeamiento del software, ingeniería de código, modelado de clases y modelado de datos.

Conclusiones parciales del capítulo

El conocimiento obtenido a partir del estudio de la arquitectura y particiones de Android permitirá definir un proceso de desarrollo de una *ROM* de manera formal, lógica y ordenada en aras de adquirir un resultado de calidad basado en la comprensión de las herramientas y conceptos fundamentales necesarios en este ámbito.

CAPÍTULO 2. PROCESO DE DESARROLLO DE *ROM* PARA ANDROID.

La modelación del proceso de desarrollo de una ROM para Android se realizó teniendo en cuenta los roles y actividades de los participantes, las decisiones a tomar y los artefactos generados que son producidos o requeridos por determinada actividad.

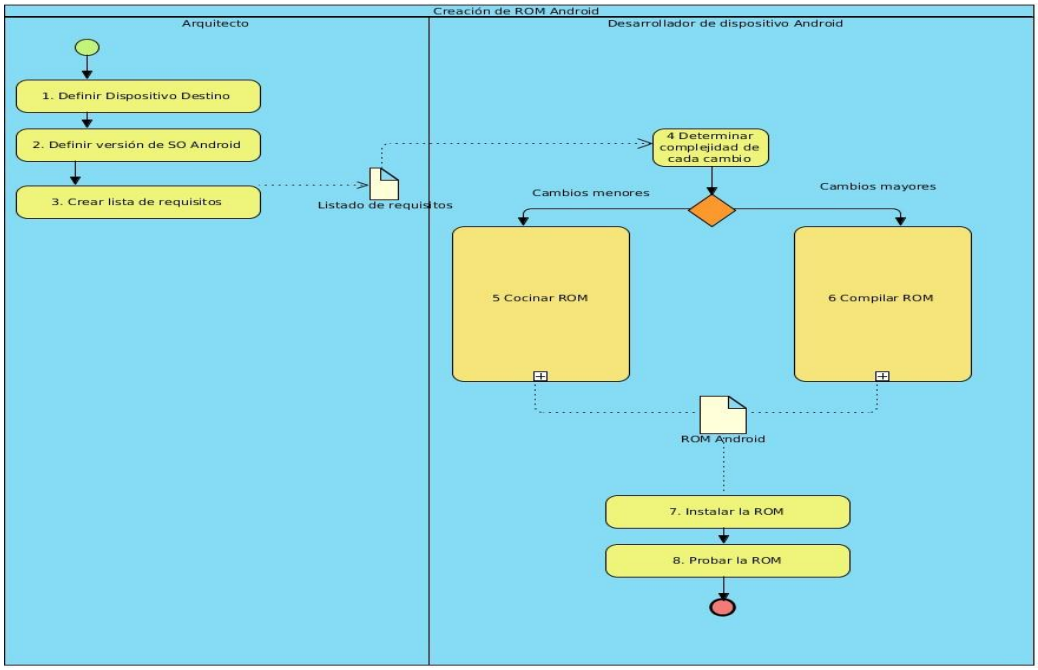


Imagen 5 Proceso de creación de *ROM* Android

2.1. Definición del dispositivo destino

Existe gran interdependencia entre la *ROM* a crear o modificar y el dispositivo en que será instalada, marcada principalmente por la implementación particular y específica de los *drivers* y núcleo compatibles con el hardware y SoC del dispositivo. Es por esto que constituye un primer paso fundamental en el proceso. Además a partir de esta selección se obtienen variables necesarias a lo largo del proceso como:

- Fabricante del dispositivo: Constituye una precondición para la selección de las herramientas a utilizar.
- SoC: Conocer qué procesador posee el terminal permite buscar homólogos para la obtención de controladores similares cuando no se cuenta con una *ROM* auténtica.

2.2. Definición de la versión de Android

Cada versión de Android posee características y requisitos de compatibilidad mínimos que han de ser cumplidos por el dispositivo destino, por lo que el paso es dependiente del dispositivo seleccionado. Para elegir la versión que será usada, deben tenerse en cuenta varios aspectos tales como la **robustez** del mismo, es decir, que el sistema contenga la capacidad y procesos de reacción apropiada ante condiciones que se encuentren fuera de su alcance. También es de vital importancia la capacidad de **preservar** los datos y ejecutar aplicaciones sin problemas de **rendimiento**.

Cada SO recibe o acepta aplicaciones específicas que deben ser compatibles. Cuanto más **actualizado** sea el sistema operativo más integración existe con las aplicaciones, mayor es el nivel de seguridad desarrollado, además, aspectos como la interfaz de usuario intensifican su madurez con cada actualización.

2.3. Creación del listado de requisitos

A continuación se procede a la selección de las modificaciones que se desean realizar a la *ROM* base o versión de Android a portar, ya sea añadir o eliminar aplicaciones por defecto, o bien, modificaciones en capas más profundas de la arquitectura del sistema para actualizar las librerías nativas de modo que se incremente la seguridad, rendimiento y estabilidad. Para esto se utiliza el modelo propuesto a continuación.

Dispositivo Destino		SoC	Versión Android
#	Tipo	Descripción	

Tabla 1 Modelo de listado de requisitos

Compuesto por:

- **Dispositivo destino**: Nombre del modelo de terminal para el que se crea la *ROM*.
- **SoC**: Identificador del SoC del dispositivo.
- **Versión Android**: Versión de OS a utilizar.
- **Tipo**: Tipo de cambio a realizar:
  - **Añadir**: Se refiere a la acción de agregar un software, característica o modificación nueva a la *ROM*.
  - **Eliminar**: Eliminar o quitar completamente una característica o aplicación de la *ROM*.
  - **Modificar**: Se refiere a realizar cambios sobre funciones o software existente en la *ROM* sin ser eliminado y reemplazado del todo.

- **Actualizar**: Tipo de modificación enfocada a solucionar errores a partir del reemplazo de librerías y funciones o la actualización total o parcial de módulos del sistema, controladores.

- **Descripción**: Describe el cambio a realizar.

2.4. Selección del modo de creación

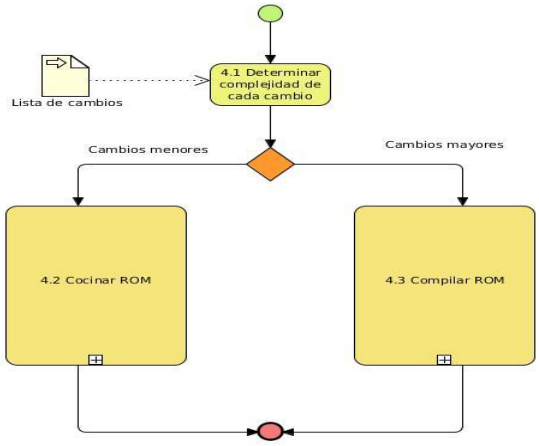


Imagen 6 Determinar modo de creación

Anteriormente se mencionaron dos métodos o formas para la obtención de una *ROM*, para determinar cuál de estos métodos debe usarse es necesario estudiar los cambios que fueron declarados y definir qué capas de la arquitectura de Android afectan. Se consideran cambios menores aquellos que implican modificaciones sobre las capas más superficiales; como la capa de Aplicaciones, además de modificaciones simples como el cambio de las animaciones de inicio, el cambio del número de compilación o modificaciones específicas al archivo *build.prop*, en estos casos debe considerarse cocinar la *ROM*. Sin embargo para decantarse por este método es necesario garantizar la existencia de un entorno de trabajo factible dado que existen cocinas específicas para cada terminal. En ocasiones no existen herramientas compatibles con el dispositivo en cuestión, sin embargo, es posible conseguir la compatibilidad con las cocinas o implementar herramientas que reproduzcan las opciones necesarias. Además es imprescindible poseer una *ROM* compatible con el dispositivo o una *ROM* perteneciente a otro modelo de igual SoC que pueda ser utilizada.

Por otra parte, para lograr un control total sobre el sistema, algo posible gracias al AOSP, es necesario compilar el código fuente. Ha de tenerse en cuenta que la compilación para dispositivos específicos requiere de los *drivers* y núcleo compatibles con la rama elegida. Además requiere la correcta configuración del entorno de trabajo.

2.5. Creación de la *ROM*

A continuación se relacionan las dos formas definidas en el documento para crear una *ROM*.

2.5.1. Cocinando ROM Android

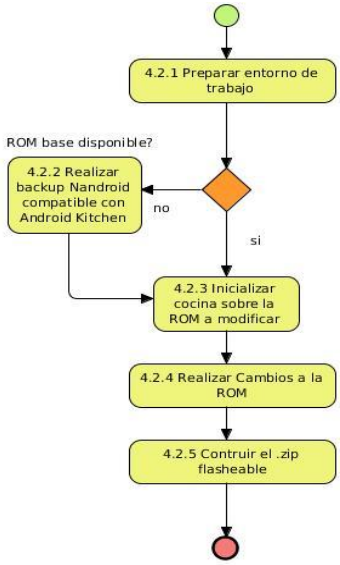


Imagen 7 Pasos para cocinar una *ROM*.

Preparar entorno de trabajo

Antes de comenzar a *"cocinar"* es necesario instalar el kit de desarrollo de Java (*JDK*) y la cocina, en este caso, *dsixda's Android Kitchen*-v-0.224. Se requiere además de un editor de texto como el Sublime Text 3.0 y una herramienta de compresión/descompresión como 7Zip.

Luego se extrae el contenido de la *ROM* en una carpeta de igual nombre para obtener los archivos *system.img* y *boot.img*, luego se mueven dichos archivos hacia la cocina, dentro de la carpeta "*original_udpate*":

```
mv system.img $KITCHEN /original_update/

mv boot.img $KITCHEN /original_update/
```

De este modo el entorno de trabajo está preparado para iniciar el proceso de personalización.

Elegir *ROM* base

Es obligatorio definir como punto de partida la *ROM* a modificar, pues sobre ésta y su estructura se erigirá el resto del trabajo, para esto además se tiene que considerar el dispositivo destino seleccionado. Suele partirse de la *Stock ROM* u otra compatible con el mismo ya que de esta forma se garantiza al menos un punto de partida válido, funcional y compatible con el terminal en cuestión. La selección de una *ROM* errónea como fuente para las modificaciones decantará en un resultado incompatible con el dispositivo destino o con deficiencias operativas.

Crear copia de seguridad Nandroid

Antes de proseguir es muy importante guardar el estado actual del dispositivo para garantizar un estado válido al cual regresar en caso de problemas. Este es precisamente el fin de las copias de seguridad Nandroid.

Para realizar la copia de seguridad se procede a *bootear* el terminal en modo de recuperación y una vez iniciado el CWM u otro sistema de recuperación avanzada se escoge la opción ***backup***. A su término el proceso habrá creado una carpeta dentro de la tarjeta SD nombrada con un formato similar a <2014-07-02.01.21.44> representando la fecha y hora de la copia de seguridad. Es recomendable hacer una copia de esta carpeta en otro dispositivo de almacenamiento para preservar los datos ante fallos en el sistema de archivos del terminal. En este punto se tiene una garantía parcial que puede servir en gran medida para recuperar el dispositivo en caso de fallos, o en caso de tener un dispositivo *brickeado*.

Por otra parte, en ocasiones puede resultar muy útil realizar la copia de seguridad para extraer los archivos pertinentes (*system.img* y *boot.img*) y utilizarlos como *ROM* base cuando no se posee la *Stock ROM* del terminal.

Realizar cambios a la *ROM*

A continuación se muestran las opciones disponibles en la versión 0.224 de la cocina Android Kitchen. Es necesario recalcar el cuidado a tener respecto a los aspectos que se modifican ya que el terminal puede resultar dañado irreparablemente.

Para abrir la cocina es necesario navegar hasta la carpeta *$Kitchen* (carpeta local donde se tiene la cocina descomprimida) y luego ejecutar:

```
./menu
```

De esta manera se inicia la herramienta Android Kitchen que muestra las siguientes opciones:

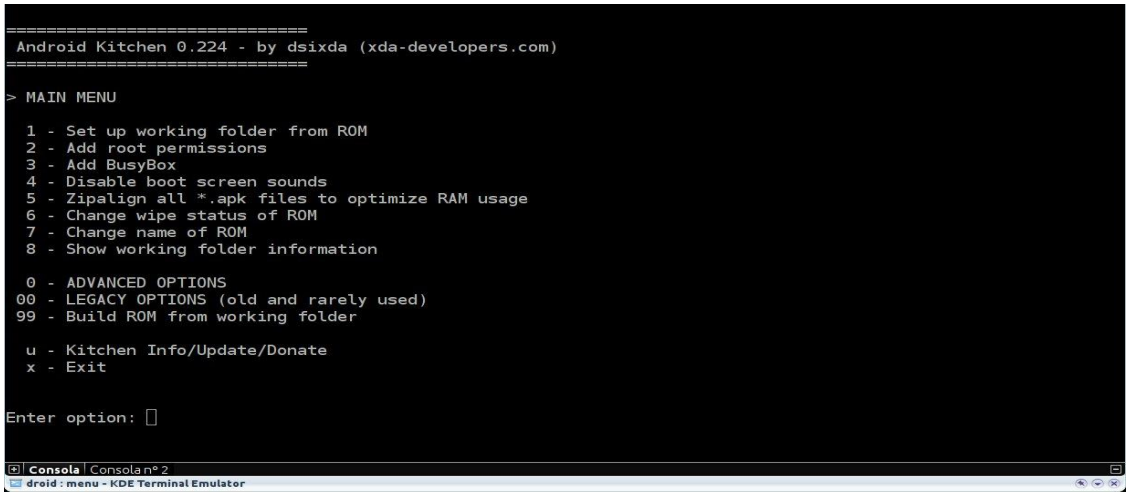


Imagen 8 Menú principal de Android Kitchen

El documento anexo #6 muestra una lista de las opciones disponibles en la herramienta.

Quitar lo que sobra "Litear"

Al cocinar una *ROM*, es posible eliminar las aplicaciones que afectan el rendimiento del dispositivo y no obedecen a los intereses del usuario, una opción que sin dudas proporciona valor añadido.

Dicho proceso es conocido como "*Litear*", proveniente de la palabra inglesa "lite" (pequeño en español), es un aforismo que se refiere a la acción de hacer algo más pequeño, en el caso abarcado se refiere a eliminar aplicaciones para reducir el tamaño final de la *ROM*. Para garantizar que la eliminación de un paquete .apk no afecte el funcionamiento del resto se utilizan herramientas como Titanium Backup, una aplicación diseñada para hacer copias de seguridad y que permite "congelar" una aplicación para probar la integridad del sistema. Tras comprobar que un paquete .apk no es vital para el funcionamiento del dispositivo o no existen dependencias sobre el mismo, éste puede ser eliminado, por lo que se procede a eliminar directamente los archivos seleccionados en:

```
$KITCHEN/WORKING_<xxxx>_<xxxx>/system/app/
```

Añadir lo nuevo

Para añadir aplicaciones directamente en la carpeta */system/app/* se deben tener en cuenta las restricciones de seguridad que impone Android sobre las firmas de las aplicaciones. Cuando se inserta una aplicación nueva en dicha carpeta, tiene que estar firmada con la misma firma digital que el resto de las aplicaciones que se

encuentran ya en el sistema, de esta manera se garantiza que la *ROM* pueda ser sólo modificada por su creador que es quien define y posee la firma utilizada. Sin embargo es posible firmar todas las aplicaciones con una nueva a través de la cocina (opción avanzada # 16) o añadir las aplicaciones nuevas en la carpeta */data/app* (opción avanzada # 13).

Build.prop

Existen otras modificaciones que pueden ser realizadas con la ayuda de una cocina, tal es el caso de las operaciones que se realizan sobre el archivo *Build.prop*.

Build.prop es un archivo de sistema generado por Android que contempla principalmente configuraciones del dispositivo. Mediante modificaciones a dicho archivo se pueden obtener mejoras de rendimiento, ahorro de la batería, aumento de la calidad de las fotografías y vídeos, entre otros.

El archivo *build.prop*, se encuentra en:

```
$KITCHEN/WORKING_<xxxx>_<xxxx>/system/build.prop
```

Para editarlo, debe utilizarse Notepad++ o Sublime Text 3.0 ya que otros editores podrían insertar caracteres extraños que corrompen el archivo y Android es altamente sensible a las modificaciones de este tipo. Ha de tenerse en cuenta que los errores con número entre cero y diez se refieren precisamente a errores sintácticos en la estructura de este fichero por lo que el proceso de instalación de la *ROM* se verá comprometido.

Por ejemplo, para cambiar el identificador de compilación de la *ROM* se modifica la línea 4 quedando de la siguiente forma:

```
ro.build.display.id=CUBADROID 1.0 (4.4-R11)
```

Dicha línea contiene el nombre que se mostrará en el menú “Acerca de” dentro de “Ajustes“. Aquí debe ponerse el nombre de la *ROM* y la compilación a la que corresponde la base. Esta operación además puede ser realizada fácilmente a través de la **opción #7** de la cocina.

En el documento anexo #7 se relaciona una lista de posibles mejoras que permite este archivo. Muchas de las alteraciones son tomadas de otras *ROMs* similares, por ende, son experimentales y han de ser comprobadas. Existen muchos conjuntos de modificaciones que tratan de resolver u optimizar diferentes

procesos en los terminales, pero al ser creados por usuarios generalmente poseen errores o no son totalmente generalizables por lo que generan inconsistencia en la *ROM*.

Cambiar el núcleo

Cambiar el núcleo por otro modificado permite aprovechar más el dispositivo, pues se aumenta el rendimiento y la duración de la batería. Este procedimiento es sencillo, pero debe ser respetado el orden correcto de dicha modificación, ya que en caso contrario podría tener efectos no deseados en el funcionamiento del dispositivo.

El primer paso para cambiar el núcleo es descomprimir el archivo que lo contiene, lo que devolverá:

- META-INF
- boot.ini

Seguidamente debe moverse el archivo **boot.ini** al directorio de trabajo, reemplazando así el núcleo anterior.

Modificaciones en feature.xml

Este archivo permite añadir algunas opciones extras a las diferentes aplicaciones del terminal, por ejemplo, añadir el botón “salir” al navegador, botón de llamada rápida a la lista de contactos, activar o desactivar el sonido del obturador, cambiar los colores de los días del calendario, informe de recepción de mensajes, etc.

Para agregar una característica se debe buscar el archivo feature.xml:

```
system/csc/feature.xml
```

Si no existe debe ser recuperado e insertarlo en la ruta indicada. A continuación se añade la línea correspondiente a la característica dentro del apartado oportuno, por ejemplo:

Característica	Línea insertada
Activar la opción “Salir” en el navegador por defecto	*<CscFeature_Web_AddOptionToTerminate>true</CscFeature_Web_AddOptionToTerminate>*
Opción “Sonido del obturador” en la aplicación de la cámara	*<CscFeature_Camera_ShutterSoundMenu>true</CscFeature_Camera_ShutterSoundMenu>*

Evitar que los SMS se conviertan automáticamente a MMS	*<CscFeature_Message_DisableConvertingEffectBetweenSMSMMS>true</CscFeature_Message_DisableConvertingEffectBetweenSMSMMS>*
Habilitar un botón para llamar en la lista de contactos	*<CscFeature_Contact_EnableCallButtonInList>true</CscFeature_Contact_EnableCallButtonInList>*
Aumentar el número máximo de contactos a 999	*<CscFeature_Contact_SetLinkCountMaxAs>999</CscFeature_Contact_SetLinkCountMaxAs>*

Tabla 2 Modificaciones en feature.xml

Construir el .zip flasheable

Para construir el .zip *flasheable* desde la cocina se introduce la opción:

99. Build ROM from working folder

Dicha opción se compone de varios pasos:

1. Alinear los paquetes para optimizar la RAM.
2. Convertir el archivo *update-script* a *updater-script*.
3. Construir el archivo update.zip.
4. Firmar el archivo .zip.
5. Renombrar el archivo resultante.

Esta opción tiene varias modalidades que permiten ejecutar las opciones listadas de manera automática, semiautomática o manual en dependencia de la experiencia y necesidades del usuario.

2.5.2. Compilación del código fuente de Android

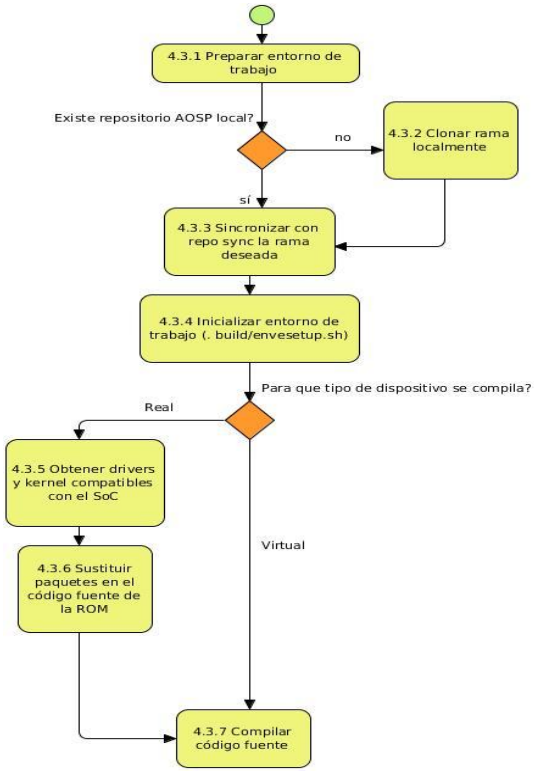


Imagen 9 Proceso de compilación de *ROM*

A diferencia del procedimiento expuesto anteriormente, compilar Android permite un control total sobre el producto final y los cambios a realizar. Es posible cualquier modificación sin importar capa o complejidad, sin embargo, esto requiere de conocimientos más profundos en el tema y de una serie de pre condiciones sobre todo en el caso de compilar para dispositivos específicos.

Una *ROM* de Android está compuesta por:

- Una plataforma móvil que sustenta las funciones básicas del dispositivo, llamada comúnmente AOSP.
- El paquete propietario de aplicaciones de Google (opcional).
- El conjunto de *drivers* y el núcleo propios del dispositivo que hacen compatible al mismo con el resto del sistema.
- Las aplicaciones.

Es por esto que compilar Android para un dispositivo requiere de los *drivers* y el núcleo del mismo. Estos componentes son creados única y exclusivamente por los fabricantes del hardware, por lo que el proceso está

indiscutiblemente atado a estos. En ocasiones existe la posibilidad de extraer *drivers* de dispositivos diferentes que compartan hardware o modelos con características similares, pero es un proceso experimental y por tanto su efectividad no está garantizada.

No obstante el SDK de Android brinda herramientas de desarrollo que potencian la creación y prueba de *ROMs*, tal es el caso del emulador que incluye. Gracias a esta herramienta es posible probar las modificaciones realizadas a una *ROM* emulando su comportamiento en las conocidas AVD (*Android Virtual Device*). Esta opción es de gran importancia pues no es necesario comprometer la integridad de un terminal para comprobar cambios en la *ROM*.

Preparar entorno de trabajo

El proceso de compilación de Android es rutinariamente probado en versiones recientes de Ubuntu LTS (12.04), pero la mayoría de las distribuciones deben poseer las herramientas de compilación requeridas.

Es necesario para poder realizar el proceso de compilación de Android garantizar los siguientes requerimientos:

- Un sistema Mac o Linux, aunque es posible también en una máquina virtual en sistemas nos soportados como *Windows*. Si se ejecuta Linux como una máquina virtual, es necesario al menos 16 GB de RAM/swap y 50 GB o más de espacio en el disco duro para poder construir el árbol de Android.
- Se requiere de un entorno de 64-bits para Gingerbread (2.3.x) o versiones superiores, incluyendo la rama máster. Versiones anteriores pueden ser compiladas en un sistema de 32-bits.
- Al menos 50 GB de espacio disponible en el disco duro para realizar un *checkout*, 100 Gb para una compilación única y 150 GB o más para múltiples compilaciones. En caso de usar *ccache*, se necesitará incluso más espacio.
- Python 2.6 - 2.7.
- GNU Make 3.81 - 3.82.
- JDK 7 para construir la rama máster de Android en el AOSP; JDK 6 para compilar desde Gingerbread hasta KitKat; JDK 5 para versiones desde Cupcake hasta Froyo.
- Git 1.7 o superior.

Escoger Rama

Algunos de los requerimientos para el entorno de compilación están determinados por la versión de Android que se planea compilar. Debe consultarse los números de compilación para obtener una lista completa de ramas que pueden ser elegidas. También se puede escoger descargar y compilar el último código fuente disponible (llamado máster), para este caso se omite la especificación de rama en el momento de inicialización del repositorio.

Inicializar

Se inicializa el entorno con el script *envsetup.sh*. Es posible reemplazar "*source*" por un punto simple para ahorrar algunos caracteres y la forma corta es más usada en la documentación oficial.

```
$ source build/envsetup.sh
```

Selección de objetivo

Se escoge que objetivo compilar con el comando *lunch*. La configuración exacta puede ser pasada como un argumento, por ejemplo:

```
$ lunch aosp_arm-eng
```

Este ejemplo se refiere a una compilación completa para emular con las opciones de debugueo habilitadas.

Si se ejecuta sin argumentos, lunch le pedirá escoger un objetivo desde el menú.

Todos los objetivos tienen la forma *BUILD-BUILDTYPE*, donde *BUILD* es un nombre codificado que se refiere a una combinación particular de características. Aquí esta una lista parcial:

Nombre de compilación	Dispositivo	Notas
aosp_arm	ARM emulator	AOSP, completamente configurado con todos los lenguajes, aplicaciones y métodos de entrada.
aosp_maguro	maguro	La versión de AOSP que corre actualmente en el Galaxy Nexus GSM/HSPA+ ("maguro").
aosp_panda	panda	La versión de AOSP que corre en el PandaBoard ("panda").

Tabla 3 3Selección del objetivo BUILD

Compilando el código

Es el momento de compilar usando *make*. *Gnu make* puede ejecutar tareas en paralelo con el argumento *–jN*, y es común usar un número de tareas N que está entre 1 y 2 veces el número de hilos de hardware de la computadora siendo usados por la compilación. Por ejemplo, en un ordenador dual-E5520 (2 CPUs, 4 núcleos con 2 hilos por núcleo), la compilación más rápida se logra con comandos entre *make –j16* y *make –j32*.

2.6. Instalación de la *ROM*

Instalar una *ROM* en un terminal Android no es un proceso complicado, aunque debe ser ejecutado con cuidado pues el dispositivo puede sufrir daños irreversibles. Dicho proceso suele variar entre dispositivos, aunque de manera general el proceso es similar al descrito a continuación.

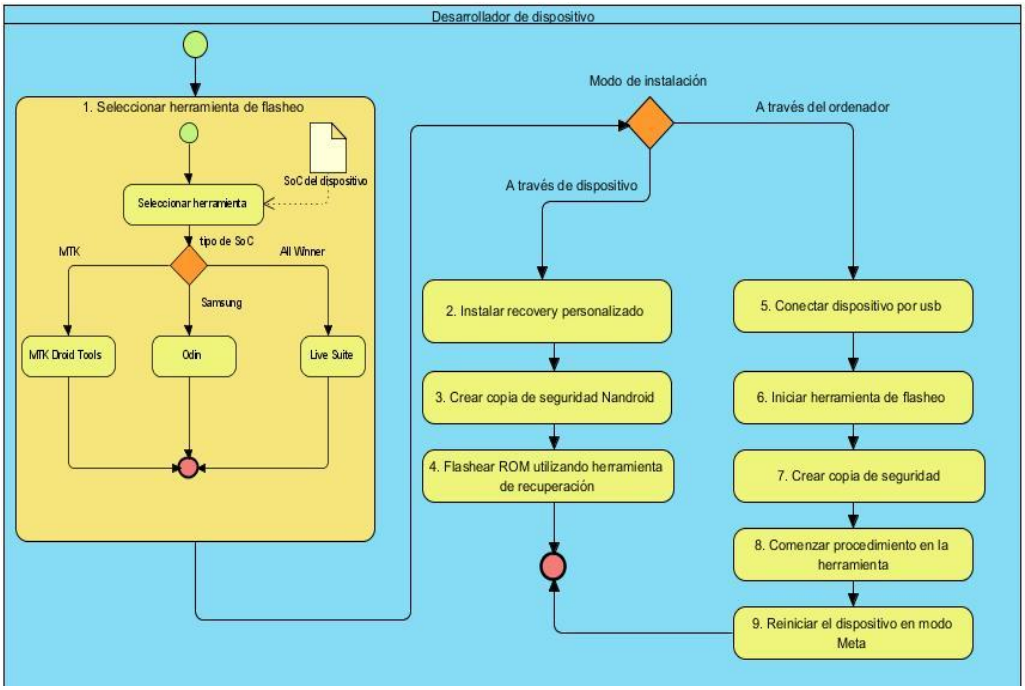


Imagen 10 Proceso de instalación de *ROM* Android

Seleccionar herramienta de instalación

Para instalar el terminal es necesario disponer de la herramienta apropiada. Generalmente estas herramientas son desarrolladas por los fabricantes del hardware y por ende específicas para cada caso.

Instalar *recovery* personalizado

Una vez seleccionada la herramienta de *flasheo* se procede a instalar el sistema de recuperación personalizado, la elección de la herramienta depende de su soporte para el terminal sobre el que se esté trabajando.

Crear copia de seguridad Nandroid

En este paso se procede simplemente a *bootear* el terminal en modo de recuperación. Es recomendable hacer una copia de esta carpeta en otro dispositivo de almacenamiento u ordenador para preservar la copia de seguridad ante fallos en el sistema de archivos del terminal. En este punto se tiene una garantía parcial que puede servir en gran medida para recuperar el dispositivo en caso de fallos por algún motivo.

Instalar *ROM*

La instalación de la *ROM* puede llevarse a cabo de diversas maneras, algunas partiendo del sistema de recuperación instalada y otras interactuando directamente a través de un ordenador y herramientas adecuadas. La selección de la técnica a utilizar está dada fundamentalmente por las condiciones del dispositivo. Usualmente se prefiere la utilización del sistema de recuperación debido a que la operación de *flasheo* es extremadamente delicada y en caso de realizarse con la ayuda de un ordenador existe el riesgo de pérdida de información a causa de problemas con el cableado y por ende todo el proceso se verá comprometido, afectando significativamente el terminal.

Conectar dispositivo a través del puerto USB

Para realizar la instalación mediante un ordenador es obligatorio utilizar la conexión USB del terminal en cuestión, por el momento no se han desarrollado modos de instalación mediante redes inalámbricas u otros medios.

Iniciar herramienta de instalación

Generalmente la herramienta escogida necesita ser previamente instalada y requiere en ocasiones de la instalación de *drivers* especiales para la comunicación entre el dispositivo y los ordenadores en los modos implementados por la misma. Luego de instalada se inicia y se suministra el archivo de dispersión específico del dispositivo.

Crear copia de seguridad

Se procede entonces a realizar una copia de seguridad del estado actual del terminal. En caso de que la herramienta no cuente con esta funcionalidad se recomienda instalar una herramienta de recuperación avanzada y realizar un *Nandroid Backup* antes de proseguir.

Comenzar procedimiento en la herramienta

Una vez seleccionados el/los archivos a *flashear* en el terminal, se inicia el procedimiento. La herramienta pausará su funcionamiento hasta detectar un dispositivo conectado en modo meta para comenzar a escribir la información en las particiones especificadas.

Reiniciar el dispositivo en modo meta

El procedimiento a seguir para iniciar un dispositivo en modo meta varía entre fabricantes, principalmente se basa en combinaciones de teclas siguiendo las órdenes específicas brindadas por el proveedor. Existen dispositivos que reinician automáticamente en modos de comunicación especializados como los terminales MTK, por lo que bastaría con reiniciar usando el comando:

```
adb reboot
```

Conclusiones parciales del capítulo

En este capítulo se detallaron los pasos del proceso para el desarrollo y mantenimiento de *ROMs* Android, donde quedaron definidas las tecnologías y herramientas adecuadas para cada una de las vertientes de la solución propuesta, definiéndose el orden lógico a seguir, recalcando en la importancia y riesgos que conllevan los subprocesos de *flasheo* en dispositivos móviles.

CAPÍTULO 3. **IMPLEMENTACIÓN Y PRUEBA DE *ROM* ANDROID**

En esta etapa se realiza la validación de la propuesta para el desarrollo y mantenimiento de una *ROM* Android. Con este fin se emplea como estrategia, la implementación de un caso de estudio práctico para los dispositivos Nut Neko - SoC MT6582, Alcatel One Touch – SoC MT6577 y para un *AVD* de Android, además de una encuesta aplicada a expertos en el tema, junto a la implementación del método experimental para validar las variables seguridad, soberanía tecnológica, socio-adaptabilidad y sostenibilidad.

3.1. Compilación para un Dispositivo Virtual Android (AVD).

Antes del despliegue de una *ROM* es imprescindible comprobar su funcionamiento, con este fin pueden ser utilizados los *AVDs* que no son más que herramientas virtuales que permiten emular un dispositivo Android. Con este objetivo se crea mediante la compilación del código fuente de Android una *ROM* evidenciando la veracidad del proceso descrito. A continuación se siguen los pasos de la solución propuesta.

1- Definir dispositivo destino.

El dispositivo destino será un emulador virtual con las siguientes características: 512 MB de RAM y tamaño pantalla de 1024 X 768.

2- Definir versión de Android.

Se decidió trabajar sobre Android 5.0 por ser la versión más reciente del sistema y se posee el repositorio del código fuente en la Universidad.

3- Crear lista de requisitos.

Según la encuesta realizada el listado de requisitos quedó de la siguiente forma:

Dispositivo Destino	SoC	Versión Android
AVD Nexus	Qualcomm Snapdragon S4	5.0

#	Tipo	Descripción
1	Eliminar	Chrome.apk (Reemplazo por Firefox)
2	Eliminar	Exchange.apk
3	Eliminar	Gmail2.apk

4	Eliminar	Music.apk
5	Eliminar	SmartCardService.apk
6	Eliminar	TvSettings.apk
7	Eliminar	UnifiedEmail.apk
8	Eliminar	GoogleCalendarSyncAdapter.apk
9	Eliminar	YouTube.apk
10	Eliminar	FileManager.apk (Reemplazo por ES File Explorer)
11	Modificar	Cambiar nombre de la *ROM* por: CubaDroid 1.0 (5.0)
12	Añadir	FNM.apk
13	Añadir	*99.apk
14	Añadir	ucintlm.apk
15	Añadir	OsmAnd.apk
16	Añadir	My Tracks.apk
17	Añadir	ES File Explorer.apk
18	Añadir	VLC.apk
19	Añadir	WPS Office Suite 6.0.apk
20	Añadir	MozillaFirefox.apk

Tabla 4 Listado de cambios para AVD

4- Determinar el modo de creación.

Se compilará el código fuente de Android para emulador siguiendo los pasos preestablecidos.

4.1- Preparar el entorno de trabajo.

Se utilizó Ubuntu 14.04 X64, en un ordenador con 4 Gb de RAM, microprocesador Core I3 (1300 GHz) y 60 Gb de espacio de almacenamiento disponible.

Instalando paquetes requeridos (Ubuntu 14.04)

Se instalaron los paquetes Python 2.7, GNU Make 3.82, JDK 7 y Git 1.7, según lo orientado por la guía oficial de AOSP. Además los necesarios para compilar específicamente en Ubuntu 14.04, ejecutando el comando:

```
$ sudo apt-get install bison g++-multilib git gperf libxml2-utils
```

4.3 Inicializar entorno de trabajo.

Para inicializar el entorno es necesario dirigirse a la carpeta donde se tiene el repositorio de AOSP y ejecutar el comando:

```
. build/envsetup.sh
```

4.4 Compilar el código fuente

Posteriormente se procede a seleccionar el tipo de dispositivo para el que se compila, en este caso se seleccionó la configuración aosp_arm-eng indicando que el objetivo es un emulador *ARM* con configuraciones adicionales para *debugueo* y desarrollo además de las aplicaciones base instalas, esto se logra ejecutando:

```
lunch aosp_arm-eng
```

Luego se comienza la compilación utilizando cuatro hilos para el proceso con el comando:

```
make -j4
```

Demoró un total de 14 horas el término del proceso.

5- Instalar la *ROM*

En el caso de las *ROMs* para emuladores no es necesaria la instalación, sólo asociarla con el dispositivo virtual deseado. Esto ocurre automáticamente cuando se compila el código fuente de AOSP.

6- Probar la *ROM*

Para iniciar el dispositivo virtual con la *ROM* compilada se ejecuta el comando:

```
emulator
```

Durante el proceso de *booteo* del AVD puede percatarse del cambio de la animación de inicio. Una vez completado se llega a la pantalla principal de la *ROM* si el proceso finalizó con éxito.

3.2. Implementación de caso de estudio práctico #1 (Nut Neko - SoC MT6582).

La empresa industrial para la informática, las comunicaciones y la electrónica, GEDEME, perteneciente al Grupo de la Electrónica; se dedica a desarrollar, producir, comercializar de forma mayorista y brindar servicios técnicos a productos de las tecnologías de la información, las telecomunicaciones y la electrónica, con destino al mercado nacional y la exportación. Entre las nuevas tareas del año en curso, la empresa se propone comenzar el ensamblado de dispositivos móviles de gama alta y medio-alta para su comercialización inicialmente en el país partiendo del modelo Nut Neko con SoC MT6582 como caso de pruebas inicial. Sin embargo dichos terminales cuentan con una *ROM* instalada que imposibilita el control de los niveles de Seguridad, Socio-Adaptabilidad, Soberanía y Sostenibilidad sobre el producto a crear.

Para esto la empresa se apoya en la Universidad de las Ciencias Informáticas con el fin de crear una *ROM* cubana que permita la modificación de los parámetros mencionados.

Para implementar la solución se seguirá el procedimiento definido previamente:

1- Definir dispositivo destino.

Como dispositivo destino se tiene el terminal Nut Neko - SoC MT6582, del fabricante MediaTek.

2- Definir versión de Android.

Se decide actualizar el dispositivo a Android 4.4.2 Kit Kat, partiendo de una *ROM* compatible de otro dispositivo con igual SoC. No es posible utilizar la versión 5.0 de Android pues aún no está disponible para este terminal.

3- Crear lista de requisitos.

Es importante hacer un estudio del consumo de recursos de las aplicaciones en el sistema, para esto existen varias opciones. Es posible a través del *ADB* conectarse al *shell* del dispositivo y listar los procesos ordenados por consumo tanto de RAM como de CPU para construir una lista de posibles candidatos a eliminar. Para esto se ejecuta en un terminal con el dispositivo conectado vía *usb*:

```
adb shell

top –s cpu # Para ordenar según consumo de cpu

top –s rss # Para ordenar según consumo de ram
```

También es posible realizar un estudio similar mediante herramientas propias de Android. Tal es el caso del administrador de aplicaciones del sistema.

Para conocer qué aplicaciones podrían mejorar los niveles de socio-adaptabilidad en el entorno actual cubano, se realizó una encuesta en la Universidad que a la fecha posee un total de 3958 lecturas y 63 comentarios por parte de los miembros de la comunidad de software libre, no solo de la Universidad de Ciencias Informáticas (UCI), sino del país en general.

En base a los resultados se obtuvo una lista de cambios deseados, en cuanto a aplicaciones, que se muestra a continuación:

Dispositivo Destino	SoC	Versión Android
Nut Neko	MT6582	4.4

#	Tipo	Descripción
1	Eliminar	Chrome.apk (Reemplazo por Firefox)
2	Eliminar	Exchange.apk
3	Eliminar	Gmail2.apk
4	Eliminar	Music.apk
5	Eliminar	SmartCardService.apk
6	Eliminar	TvSettings.apk
7	Eliminar	UnifiedEmail.apk
8	Eliminar	GoogleCalendarSyncAdapter.apk
9	Eliminar	YouTube.apk
10	Eliminar	FileManager.apk (Reemplazo por ES File Explorer)
11	Actualizar	Actualizar librería glibc para eliminar el bug GHOST
12	Modificar	Cambiar nombre de la *ROM* por: CubaDroid 1.0 (4.4 R11)
13	Añadir	FNM.apk
14	Añadir	*99.apk
15	Añadir	UCIntlm.apk
16	Añadir	OsmAnd.apk
17	Añadir	My Tracks.apk
18	Añadir	ES File Explorer.apk

19	Añadir	VLC.apk
20	Añadir	WPS Office Suite 6.0.apk
21	Añadir	MozillaFirefox.apk

Tabla 5 Listado de cambios

4- Determinar modo de creación.

Cocinando *ROM* para Nut Neko

Debido a que todos los cambios son realizables con la cocina y a que la ausencia de *drivers* y núcleo imposibilita el proceso de compilación, se opta por la primera opción: Cocinar una *ROM*.

4.2 Preparar el entorno de trabajo.

A continuación se definen las herramientas, plataforma y entorno de desarrollo adecuados en la confección de una *ROM* para el dispositivo de pruebas Nut Neko, teniendo en cuenta la *ROM* base, el *SoC* del terminal, fabricante y las transformaciones a realizar.

El terminal posee un SoC fabricado por MediaTek por lo que las herramientas asociadas con el proceso ya están predefinidas. Para cocinar se utilizó un entorno Linux, específicamente Ubuntu 14.04 X64 debido a que la cocina Android Kitchen 0.224 es compatible con dicho sistema.

4.2.3 Inicializar cocina sobre la *ROM* a modificar.

Para inicializar la cocina es necesario navegar hasta la carpeta *$Kitchen* y luego ejecutar:

```
./menu
```

Luego se selecciona la opción 1 ***Set up working folder from ROM***, que crea el espacio de trabajo y desempaqueta la *ROM*.

4.2.4 Realizar cambios a la *ROM*

Para añadir las aplicaciones se realizó una copia de estas a la carpeta */app* dentro del directorio de trabajo de la cocina:

```
$KITCHEN/WORKING_<xxxx>_<xxxx>/system/app/
```

Se decidió integrar BusyBox a la *ROM* y cocinar una versión de desarrollo permitiendo el uso de librerías nativas del entorno Linux, para esto se utilizó la opción número 3 de la cocina: '*Add BusyBox'*. Por otra parte para eliminar aplicaciones es necesario cerciorarse de que no son imprescindibles para el correcto funcionamiento del sistema y no afectan dependencias de otras aplicaciones. Para esto se utilizó la herramienta Titanium Backup que permite simular el proceso de eliminación sin causar daños irreversibles sobre la *ROM* o el dispositivo.

Al acceder a dicha herramienta se observa una pantalla inicial en la que se debe pulsar el botón "Copiar/Restaurar", luego muestra una lista con todos los paquetes del sistema. Se selecciona el paquete que se desea congelar y se pulsa sobre él. Aparecerán las propiedades del paquete, y se seleccionará la opción "Congelar". A continuación aparecerá el paquete congelado con fondo azul. Si tras congelar una aplicación el sistema operativo no resultó dañado, dicha aplicación puede ser eliminada sin dificultad.

4.2.5 Reconstruir la *ROM*

Una vez realizados los cambios previstos se procede a reconstruir la *ROM* en un archivo instalable por lo general en formato .zip, compatible con las herramientas de *flasheo* tradicionales o los sistemas de recuperación avanzados. Para esto se ejecuta la opción *99. Build ROM from working folder* (construir *ROM* desde carpeta de trabajo. Dicha opción además realizará un conjunto de operaciones necesarias para lograr la compatibilidad y mejorar el rendimiento.

5- Instalar la *ROM*.

Luego se procede a instalar la *ROM* a través de la herramienta SP Flash Tool. Para esto:

- Se crea el archivo de dispersión del dispositivo. Esto se logra a través de la herramienta MTK Droid Tools siguiendo los siguientes pasos:
    - Se selecciona la opción *Blocks Map* (mapa de bloques) con el dispositivo conectado al ordenador.
    - Se selecciona la opción *Create Scatter File* (crear archivo de dispersión) con lo que se obtendrá el archivo deseado.
- Se carga el archivo de dispersión apropiado para el dispositivo a *flashear*.
- Se seleccionan las particiones a *flashear*, en este caso solamente el *system.img* creado en la partición denominada "android".

- Se reinicia el dispositivo entrando en modo *meta* de manera que el SP Flash Tool identifique el terminal e inicie el proceso de instalación a través de la consola con el dispositivo aún conectado ejecutándose:

```
adb reboot
```

Una vez terminada la instalación, se inicia el dispositivo en modo de recuperación y se limpian los *caches* del sistema y los datos del usuario para evitar problemas de compatibilidad o inconsistencias en el sistema.

3.3. Implementación de caso de estudio práctico #2 (Alcatel One Touch - SoC MTK6577).

La Universidad de las Ciencias Informáticas UCI se encuentra en estos momentos en un proceso de actualización tecnológica y tiene como una de sus directrices aprovechar las ventajas que ofrece el uso de dispositivos Android. Para esto se decidió asignar dispositivos móviles a los directivos de la Universidad. Sin embargo, es necesario aumentar los niveles de seguridad en los dispositivos antes de ser distribuidos, además de mejorar en lo posible el desempeño general de los mismos. Para esto se cuenta con el dispositivo de prueba Alcatel One Touch con SoC MTK6577.

Con el fin de lograr su objetivo la Universidad se apoya en el centro de desarrollo CESOL para la creación de una *ROM* personalizada.

Para implementar la solución se seguirá el procedimiento definido previamente:

1- Definir dispositivo destino.

Como dispositivo destino se tiene el terminal Alcatel One Touch - SoC MTK6577, del fabricante MediaTek.

2- Definir versión de Android.

Se decide mantener Android 4.0.1, partiendo de la propia *Stock ROM* del dispositivo. No es posible utilizar una versión posterior de Android pues no se cuenta con una *ROM* compatible con el terminal.

3- Crear lista de requisitos.

En este paso se realiza el mismo proceso descrito en el caso de estudio anterior por lo que no se duplicará la información, finalmente se identificó un total de sesenta y nueve aplicaciones a eliminar sin afectar el comportamiento de la *ROM*.

4- Determinar modo de creación.

La cocina, en su versión actual no cuenta con soporte para dispositivos MTK, por otra parte, no se poseen *drivers* y núcleo del terminal por lo que queda descartada la opción de compilar Android para este dispositivo.

Aunque no se cuenta con la cocina, la experiencia adquirida permite reproducir los procesos que lleva a cabo la herramienta con las particularidades impuestas por el fabricante MTK. Es por esto que se decide cocinar la *ROM*.

Cocinando *ROM* para Alcatel One Touch

4.2 Preparar el entorno de trabajo.

Se utiliza de igual manera Ubuntu 14.04 y las herramientas nativas del sistema para compresión/descompresión.

4.2.3 Inicializar cocina sobre la *ROM* a modificar.

Para trabajar con el archivo system.img se creó un *script* (*systool.sh*) que permite abrir o montar el *system.img* en una carpeta, realizar los cambios y luego cerrar nuevamente el *.img* para su uso posterior. A continuación se muestra el *script*:

```
#!/bin/bash
# Creado por David Alejandro Reyes Milian & Natalí Martínez Sarduy
#USO: systool abrir|cerrar $ruta_img
function startWorkSpace(){
	if [[ ! -d $SYSTEM_DIR ]]; then
		echo "Creando $SYSTEM_DIR"
		mkdir $SYSTEM_DIR
	else
		echo "Existe $SYSTEM_DIR"
		umount $SYSTEM_DIR
	fi
}
SYSTEM_DIR="./system"
if [[ $1 == "abrir" ]]; then
	echo "Montando $2"
```

```
        startWorkSpace
        mount -t ext4 -o loop $2 $SYSTEM_DIR
fi
if [[ $1 = "cerrar" ]]; then
        echo "Desmontando $2"
        umount $SYSTEM_DIR
        echo "Desmontado $2"
        if [[ -d $SYSTEM_DIR ]]; then
                rmdir $SYSTEM_DIR
                echo "$SYSTEM_DIR eliminado"
        fi
fi
```

Con el uso de este *script* se abre la imagen ejecutando:

```
./systool abrir ruta_imagen
```

Obteniendo la imagen abierta en la carpeta *./system.*

4.2.4 Realizar cambios a la *ROM*

En el caso del Alcatel One Touch existen particularidades en cuanto a la distribución de las aplicaciones introducidas por el desarrollador original de la *ROM*. Existen dos locaciones principales donde se encuentran las aplicaciones que conforman la *ROM*:

- */system/app*: Aplicaciones del sistema, intocables por el usuario final en tiempo de ejecución.
- */custpack*:
    - *removeable*: Aplicaciones que son totalmente eliminables por el usuario
    - *unremoveable*: Aplicaciones bloqueadas al usuario, no eliminables.

Como se puede apreciar, se encuentran en particiones diferentes por lo que durante el proceso de *flasheo* ha de tenerse en cuenta e instalar ambas particiones en el dispositivo.

Para añadir las aplicaciones se realizó una copia de estas a las carpetas mencionadas según los intereses de desarrollo dentro del directorio de trabajo de la cocina.

4.2.5 Reconstruir la *ROM*

Una vez realizados los cambios previstos se procede a reconstruir la *ROM* en un archivo instalable *system.img*. Esto se logra ejecutando:

```
./systool cerrar
```

5- Instalar la *ROM*.

Luego se procede a instalar la *ROM* a través de la herramienta SP Flash Tool. Para esto:

- Se crea el archivo de dispersión del dispositivo.
- Se carga el archivo de dispersión apropiado para el dispositivo a *flashear*.
- Se seleccionan las particiones a *flashear*, en este caso el system.img y custpack.img creados en las particiones “android” y “custpack” respectivamente.
- Se reinicia el dispositivo entrando en modo meta de manera que el SP Flash Tool identifique el terminal e inicie el proceso de instalación a través de la consola con el dispositivo aún conectado ejecutando:

```
adb reboot
```

Una vez terminada la instalación, se inicia el dispositivo en modo de recuperación y se limpian los *caches* del sistema y los datos del usuario para evitar problemas de compatibilidad o inconsistencias en el sistema.

3.4. Evaluación de la propuesta de solución

Como parte de la validación de la propuesta es necesario evaluar la completitud y veracidad de la misma a partir de variables e indicadores asociados con el objetivo a cumplir. Para esto se utilizó el método experimental.

3.4.1. Método experimental

Tipo de método de investigación en el que el investigador controla deliberadamente las variables para delimitar relaciones entre ellas, está basado en la metodología científica. En este método se recopilan datos para comparar las mediciones de comportamiento de un grupo control, con las mediciones de un grupo

experimental. Las variables que se utilizan pueden ser variables dependientes o variables independientes (las que el investigador manipula para ver la relación con la dependiente) (bloglosariopsa, 2008).

3.4.2. Aplicación del método

Tras un estudio detallado del problema en cuestión, así como de sus aristas fundamentales se concluye que las dificultades están relacionadas fundamentalmente con deficiencias y riesgos en la seguridad, falta de control sobre las aplicaciones en Android, afectándose la soberanía tecnológica y la socio adaptabilidad, así como con el desconocimiento total o parcial en el tema, traducido en la improductividad y pérdida de oportunidades en el mercado actual, convirtiéndose en una fuente de perdidas, por ende, no sostenible. Para verificar las variables expuestas se definen los indicadores pertinentes.

Variable # 1: seguridad

Para medir la presente variable se definieron diferentes indicadores:

1. Nivel de acceso de aplicaciones a redes y servicios.

2. Vulnerabilidades de librerías internas de la distribución de Android.

3. Cantidad de permisos sospechosos en aplicaciones instaladas.

Variable # 2: socio adaptabilidad

1. Cantidad de aplicaciones útiles en el contexto cubano.

2. Cantidad de aplicaciones cubanas en la *ROM*.

3. Cantidad de aplicaciones inutilizables en el contexto cubano.

4. Soporte brindado a dispositivos ensamblados en Cuba.

Variable # 3: soberanía tecnológica

1. Poder de decisión sobre cambios en la *ROM*.

2. Nivel de complejidad de cambios realizables.

Variable # 4: sostenibilidad

1. Nivel de conocimientos en el tema.

2. Posibilidades de introducción de la economía cubana en el mercado.

3.4.2.1. Aplicación del método en cuanto a la variable seguridad.

Android, a grandes rasgos, tiene un mecanismo de seguridad bastante eficaz, enfocado en facilitar la vida a desarrolladores y usuarios a la vez. Para que una aplicación tenga acceso a algún servicio del dispositivo, éste tiene que ser declarado previamente por el desarrollador, luego durante el proceso de instalación en el dispositivo, el usuario tiene que aceptar los permisos que requiere dicha aplicación de modo que queda protegido. Sin embargo, el usuario promedio no presta atención a este evento tan importante induciendo así un sin número de problemas de seguridad. Además existen muchas aplicaciones que no justifican los permisos que requieren.

Anteriormente a la realización de la presente investigación se desconocía la forma en la que ocurren los procesos de comunicación entre los dispositivos y el resto de la red y servicios, permitiendo la ocurrencia de fugas o robos de información. Se desconocía además la forma de actualizar la versión de Android en una *ROM* para un dispositivo en específico por lo que las actualizaciones y parches de seguridad eran inaccesibles. Por otra parte no existía un método o forma mediante la cual se pudieran reemplazar librerías nativas de la *ROM* posibilitando la encriptación de llamadas a bajo nivel, u otros mecanismos de protección y seguridad implementados por la propia institución.

La creación de un proceso de desarrollo y mantenimiento de *ROMs* Android permite controlar las aplicaciones instaladas en los dispositivos, así como los permisos que requieren y el nivel de acceso de estas a redes y servicios. Es posible con la solución propuesta abarcar todas las deficiencias encontradas y ampliar los niveles de seguridad en los dispositivos. Además del hecho de que se cuenta con la posibilidad de compilar Android a partir del código fuente abierto por lo que todas las librerías y funcionalidades del sistema pueden ser estudiadas y modificadas según lo amerite el caso.

3.4.2.2. Aplicación del método en cuanto a socio-adaptabilidad.

Los dispositivos móviles son distribuidos con una *ROM* instalada por el proveedor desde la fábrica. Inicialmente dicha *ROM* posee una variedad de aplicaciones tales como: Facebook, Ebay, Twitter, Gmail, Google Talk, You Tube, Google play, etc. Muchas de esas aplicaciones, o bien obedecen a necesidades ajenas al usuario cubano o no son accesibles desde Cuba. Por lo que el usuario en Cuba estaba confinado a tener estas aplicaciones instaladas sin voto alguno, afectándose además el rendimiento del dispositivo.

La personalización de *ROMs* para dispositivos móviles permite la eliminación de muchas de estas aplicaciones, aumentando considerablemente el rendimiento. Permite además la incorporación de aplicaciones creadas para satisfacer intereses y necesidades particulares del usuario cubano.

La posibilidad de soporte y mantenimiento para los dispositivos móviles es otra de las ventajas que proporciona el proceso propuesto, debido a que actualmente ningún dispositivo Android en Cuba tiene acceso a las actualizaciones que proveen los fabricantes.

El estudio actual constituye un avance en el entendimiento de los mecanismos de mantenimiento y actualización para dispositivos móviles Android ya que muchos de los aspectos abarcados se relacionan directamente con dicho proceso, por lo que puede considerarse un punto de partida para la implementación de servicios de actualización masiva y mantenimiento para terminales en el país.

3.4.2.3. Aplicación del método en cuanto a la variable soberanía tecnológica.

Anteriormente no era posible realizar modificaciones sobre la *ROM* instalada en un dispositivo Android, por lo que no se tenía poder de decisión sobre muchos de los aspectos relacionados con el tema. No era posible tomar decisiones de ningún tipo sobre el camino a seguir para las nuevas tecnologías en los dispositivos móviles, por lo que el futuro se denotaba incierto.

Sin embargo a partir del proceso expuesto como propuesta de solución se obtiene poder de decisión sobre los dispositivos Android, permitiendo realizar cambios de acuerdo a las necesidades actuales sin restricciones sobre el nivel de complejidad de los mismos.

3.4.2.4. Aplicación del método en cuanto a la variable sostenible.

Actualmente Cuba asimila una penetración tecnológica en lo que a telefonía móvil se refiere, siendo Android el principal exponente, además se evidencia el auge de las comunidades de desarrollo de software libre en el país. Sin embargo el país no cuenta con el conocimiento y herramientas necesarias para aprovechar las condiciones existentes en aras de desarrollar productos de valor comercializable que favorezcan la sostenibilidad económica.

La solución propuesta aporta un proceso y el conocimiento necesario para el desarrollo de *ROMs* para dispositivos Android logrando así un primer paso para la introducción en el mercado tecnológico emergente con proyectos sostenibles debido a la importancia y magnitud que pueden llegar a alcanzar.

3.4.3. Comparación de las variables implicadas.

No	Variable	%Antes	%Después
1	Seguridad	30	70
2	Socio-adaptabilidad	30	100
3	Soberanía tecnológica	10	70
4	Sostenibilidad	20	60

Tabla 6 Comparación de las variables antes y después del desarrollo de la solución propuesta

3.5. Encuesta a expertos

Con el objetivo de medir la completitud y veracidad de la propuesta de solución se realizó una encuesta implicando a personal experto en temas de Android y dispositivos móviles. La encuesta forma parte de los anexos del presente documento. A continuación se muestran los resultados.

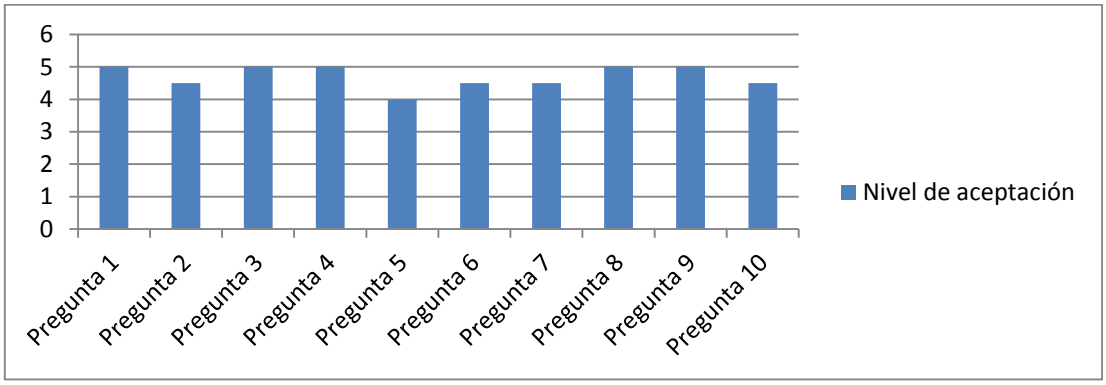


Gráfico de barras 1

Del resultado obtenido se concluyen los siguientes hechos:

- El proceso descrito permite la modificación y obtención de una *ROM* a través del código fuente de Android, 100% de aceptación.
- El proceso descrito permite la modificación y obtención de una *ROM* a través del uso de la herramienta dsixda's Android Kitchen-v-0.224(la cocina) u otras herramientas existentes o desarrolladas particularmente para el dispositivo, 90% de aceptación.
- El proceso descrito permite aumentar la seguridad en los dispositivos Android, 100% de aceptación.

- Conocer el proceso de creación y mantenimiento de *ROMs* Android permite incrementar la capacidad decisional sobre los dispositivos móviles ensamblados en Cuba, 100% de aceptación.
- Conocer el proceso de creación y mantenimiento de *ROMs* Android permite promover el desarrollo sostenible de aplicaciones basadas en Android, 80% de aceptación.
- Conocer el proceso de creación y mantenimiento de *ROMs* Android permite aumentar la seguridad en los dispositivos Android, 90% de aceptación.
- La selección de los cambios realizados en la *ROM* permite aumentar el rendimiento de los dispositivos tras eliminar las aplicaciones bloatware, 90% de aceptación.
- La selección de los cambios realizados en la *ROM* permite promover el uso de aplicaciones cubanas, 100% de aceptación.
- La selección de los cambios realizados en la *ROM* permite aumentar los niveles de socio-adaptabilidad a través de la incorporación de aplicaciones que respondan a necesidades del usuario cubano, 100% de aceptación.
- La selección de los cambios realizados en la *ROM* permite aumentar los niveles de seguridad de la información contenida en el dispositivo, 90% de aceptación.

Conclusiones parciales del capítulo

Los anteriores casos de estudio positivos avalan el proceso de creación de la *ROM* para los dispositivos estudiados llevando a la conclusión de que este pudiese replicarse e implantarse en entornos industriales lo cual favorecería a **la sostenibilidad** económica del proyecto dado que se realizarían productos de valor comercializable, a **la seguridad** ya que se tendría mayor control de las aplicaciones instaladas y los permisos que estas tienen sobre la información de los usuarios, la **socio-adaptabilidad** sería potenciada dado que se incluyen siempre las aplicaciones hechas por cubanos para cubanos y todo esto conllevaría a una mayor **soberanía tecnológica** en el proceso de elaboración de teléfonos para la población cubana.

CONCLUSIONES

El estudio del arte de las personalizaciones, así como de las tecnologías y herramientas necesarias para la creación de una *ROM*, unida a la caracterización de Android permitieron esclarecer las bases sobre las cuales se erigió la propuesta de solución.

Se creó el proceso para el desarrollo y mantenimiento de *ROMs* Android para dispositivos móviles ensamblados en Cuba, definido por siete etapas fundamentales: definición del dispositivo destino, definición de la versión de Android, creación de la lista de requisitos, evaluación de la complejidad de cada cambio, creación, instalación y prueba de la *ROM*.

La solución fue validada a través de tres casos de prueba, dos sobre los dispositivos reales Nut Neko y Alcatel One Touch y una sobre un AVD emulando el dispositivo Nexus 7. Además de la realización de una encuesta a expertos en el tema y el método experimental desarrollado para valorar la completitud de la solución propuesta.

Se cumplieron los objetivos planteados en la investigación ya que se desarrolló satisfactoriamente la solución propuesta permitiendo aumentar los niveles de sostenibilidad, seguridad, socio-adaptabilidad y soberanía tecnológica.

RECOMENDACIONES

1. Crear un repositorio de *ROMs* en la institución para potenciar el desarrollo futuro.
2. Crear una herramienta para la instalación masiva de *ROMs* en dispositivos móviles ensamblados en Cuba.
3. Continuar el estudio del proceso en aras de portar *ROMs* con futuras versiones de Android utilizando métodos genéricos.
4. Desarrollar una herramienta especializada en el proceso de cocina de *ROMs* tomando como base Android Kitchen v0.224 para dar soporte exclusivo a los dispositivos ensamblados en Cuba.

REFERENCIAS BIBLIOGRÁFICAS

Alvarado, Patricia. 2015. Ipadizate. *Ipadizate.* [En línea] 2015 de febrero de 2015. http://www.ipadizate.es/2015/02/25/ios-android-lideran-cuota-mercado-smartphones-143391/.

Android. 2013. Android. *Android.* [En línea] 2013. https://source.android.com/.

Android Developers. 2015. Android Developers. [En línea] 2015. adb. http://developer.android.com/tools/help/adb.html.

Androizados. 2014. Androizados. [En línea] diciembre de 2014. [Citado el: 10 de marzo de 2015.] http://androizados.com.

bloglosariopsa. 2008. bloglosariopsa. *bloglosariopsa.* [En línea] 2008. https://bloglosariopsa.wordpress.com/2008/11/12/metodo-experimental/.

Castro, Alberto. 2014. Computer hoy. [En línea] 30 de enero de 2014. ROM. http://computerhoy.com/.

CyanogenMod Wiki. 2014. CyanogenMod Wiki. [En línea] 2014. [Citado el: 10 de 3 de 2015.] fastboot. http://wiki.cyanogenmod.org/w/Doc:_fastboot_intro.

Fuentes, Allan Pierra. 2011. Conceptualización y Reestructuración Estratégica de la Distribución Cubana de GNU/Linux “Nova”. La Habana, Cuba : s.n., 20 de octubre de 2011.

Linfo. 2007. Linfo. [En línea] 2007. [Citado el: 10 de 3 de 2015.] root. http://www.linfo.org/root.html.

Martínez, Fali. 2012. Up to down. *Up to down.* [En línea] noviembre de 2012. http://blog.uptodown.com/las-mejores-roms-para-personalizar-nuestro-smartphone-android/.

Open Handset Alliance. 2007. Open Handset Alliance. *Open Handset Alliance.* [En línea] 11 de 2007. http://www.openhandsetalliance.com/press_110507.html.

Open Source Initiative. 2015. Open Source Initiative. [En línea] 2015. [Citado el: 10 de 3 de 2015.] http://opensource.org/docs/osd.

Riehle, Dirk. 2000. *Framework Design: A Role Modeling Approach.* 2000. framework.

Smart Mobile Phone Solutions. 2012. Smart Mobile Phone Solutions. [En línea] 2012. [Citado el: 10 de 3 de 2015.] http://www.smartmobilephonesolutions.com/content/phone-stuck-in-emergency-mode.

TeamWin. 2015. TeamWin. [En línea] 2015. [Citado el: 10 de 3 de 2015.] TWRP. http://teamw.in/project/twrp2.

Ticbeat. 2012. Ticbeat. *Ticbeat.* [En línea] 2012. las-aplicaciones-mviles-android-sospechosas-infografa. http://www.ticbeat.com/sim/las-aplicaciones-mviles-android-sospechosas-infografa/.

What is the GNU Hurd? **gnu.org. 2010.** 2010. kernel.

XDA Developers. 2015. XDA Developers. [En línea] 2015. [Citado el: 10 de 3 de 2015.] nandroid backup. http://forum.xda-developers.com/wiki/NANDroid.

ANEXOS

Anexo 1: Modelo de encuesta publicada en humanos.uci.cu.

Crear una *ROM* cubana brindará grandes beneficios en cuanto a soberanía, seguridad, sostenibilidad y socio-adaptabilidad además del número de posibilidades que puede ofrecer al desarrollo tecnológico del país. Dada la importancia del asunto y el auge actual de los dispositivos móviles, es del interés de nuestra institución conocer todas las opiniones para lograr una *ROM* personalizada, rápida y adaptada a las necesidades de los usuarios cubanos. Con este fin se proponen las siguientes encuestas para determinar las aplicaciones óptimas en nuestro contexto. Muchas de estas aplicaciones son ***Open Source*** (de código abierto), totalmente modificables por lo que son favoritas naturalmente.

LAUNCHER		**GESTOR DE ARCHIVOS**	
Nativo(por defecto del OS)		Nativo(por defecto del OS)	
Home Launcher(*OPEN SOURCE*)		Astro File Manager	
Android Launcher Plus(*OPEN SOURCE*)		Ghost Commander File manager	
Trebuchet Launcher (*OPEN SOURCE*)		File Expert	
Ubuntu Launcher(*OPEN SOURCE*)		Root Explorer	
ADW Launcher(*OPEN SOURCE*)		ES File Explorer(*OPEN SOURCE*)	
Launcher3(*OPEN SOURCE*)			
Nova Launcher			
Go Launcher			
Next Launcher			
CLIENTE DE CORREOS		**DIALER**	
Nativo(por defecto del OS)		NubDial(*OPEN SOURCE*)	
K-9 Mail(*OPEN SOURCE*)		Swipe Dialer	

Mailbox		Nativo(por defecto del OS)	
Yahoo! Mail		Contacts	
REPRODUCTORES MULTIMEDIA		**VISORES/EDITORES DE TEXTO**	
Nativo(por defecto del OS)		Nativo(por defecto del OS)	
XBMC Media Center(*OPEN SOURCE*)		CoolReader (*OPEN SOURCE*)	
Jamendo Player(*OPEN SOURCE*)		Foxit Mobile	
VLC(*OPEN SOURCE*)		OfficeSuite Pro 6	
MX Player			
NAVEGADOR WEB		**MAPAS**	
Nativo(por defecto del OS)		Nativo(por defecto del OS)	
Firefox(*OPEN SOURCE*)		OsmAnd(*OPEN SOURCE*)	
Chromium(*OPEN SOURCE*)		Locus Pro	
Chrome			
Opera Browser			
Dolphin Browser			
GESTOR DE NOTAS		**CÁMARA**	
Nativo(por defecto del OS)		Nativo(por defecto del OS)	
Evernote		Open Camera(*OPEN SOURCE*)	
Catch		Camera ZOOM FX	
Springpad		Photo Studio PRO	
Google Keep		Camera Awesome	
		Cyanogen camera	

CALENDARIO		**GPS**	
Nativo(por defecto del OS)		My Tracks(OPEN SOURCE)	
Standalone Calendar(*OPEN SOURCE*)		Avare(*OPEN SOURCE*)	
Calendar View(*OPEN SOURCE*)		Angry GPS	
Sunrise Calendar		GPS Status	
Calendar			
Business Calendar			
APKS CUBANAS		**Otras**	
FNM		Zapya	
Ecured			

Además de estas aplicaciones estamos abiertos a sugerencias de todo tipo, si alguien posee el resto de los instaladores, por favor, háganoslo llegar. Esperamos despertar un debate bastante nutrido para tomar las decisiones más acertadas en este tema novedoso.

Anexo 2: Modelo de encuesta realizada a expertos.

Datos a recoger de los encuestados:

Nombre y apellidos:

Categoría científica:

Centro de trabajo:

Área(s) de investigación:

Años de experiencia en el área:

Considere las siguientes afirmaciones y proponga una calificación entre 1 y 5, donde 1= negación absoluta y 5= validación total de la propuesta.

El proceso descrito permite:

	Marque aquí
1. La modificación y obtención de una *ROM* a través del código fuente de Android.	
2. La modificación y obtención de una *ROM* a través del uso de la herramienta dsixda's Android Kitchen-v-0.224(la cocina) u otras herramientas existentes o desarrolladas particularmente para el dispositivo.	
3. Abarcar los terminales de prueba y es generalizable.	

Conocer el proceso de creación y mantenimiento de ROMs Android permite:

	Marque aquí
4. Incrementar la capacidad decisional sobre los dispositivos móviles ensamblados en Cuba.	
5. Promover el desarrollo sostenible de aplicaciones basadas en Android.	
6. Aumentar la seguridad en los dispositivos Android.	

La selección de los cambios realizados en la ROM permite:

	Marque aquí
7. Aumentar el rendimiento de los dispositivos tras eliminar las aplicaciones bloatware.	
8. Promover el uso de aplicaciones cubanas.	
9. Aumentar los niveles de socio-adaptabilidad a través de la incorporación de aplicaciones que respondan a necesidades del usuario cubano.	
10. Aumentar los niveles de seguridad de la información contenida en el dispositivo.	

Anexo 3: Opciones del CWM

1. ***reboot system now***: Reiniciar el sistema.
2. ***update from sdcard***: Permite realizar la instalación de cualquier actualización oficial o no oficial, *ROM*, núcleo o tema que se pueda instalar desde un archivo *zip* de recuperación, o un *update.zip*, siempre y cuando el archivo sea nombrado correctamente y colocado en la raíz de la tarjeta *SD* (es decir, CWM no reconoce archivos de actualización en subcarpetas de las jerarquía de ficheros). Al seleccionar esta opción (y en la mayoría de las opciones que se describen a continuación), aparecerá una confirmación intencionada para evitar toques erróneos en otras opciones, logrando evitar problemas a causa de tecleos accidentales.
3. ***wipe data/factory reset***: Borra todos los datos de usuario, así como la memoria caché. Permite dejar el teléfono en el estado en que estaba cuando se compró o se instaló por primera vez.
4. ***wipe cache partition*****:** Limpia la partición de memoria caché para borrar todos los datos acumulados durante su uso. Se utiliza a menudo antes de instalar una nueva *ROM*, aplicación o núcleo a través del modo de recuperación.
5. ***install zip from sdcard***
    a. ***apply / sdcard / update.zip*****:** Tiene la misma función que la opción #2.
    b. ***select zip from sdcard*****:** Permite instalar cualquier archivo *.zip* (con cualquier nombre) desde la tarjeta *SD*. El archivo puede ser para una *ROM*, un núcleo, una aplicación, un tema o cualquier componente mientras esté en formato *zip-flashable* de recuperación. Esta es la opción más utilizada para la instalación de una *ROM* descargada y copiada a la tarjeta *SD*.
    c. ***toggle signature verification*****:** Activa/desactiva la verificación de firmas y apaga el dispositivo. Cuando la verificación de firmas está activada, no es posible instalar una *ROM* personalizada que no esté firmada por los desarrolladores (la mayoría no son firmadas). Apagar el dispositivo omite la comprobación de verificación de firmas y continúa con la instalación.
    d. **+++++*Go Back*+++++**: Lleva al menú principal.
6. ***backup and restore*****:** Una de las características más importantes que ofrece el sistema de recuperación personalizado, también conocido como copia de seguridad *Nandroid*, permite tomar una instantánea de toda la memoria interna del teléfono, incluyendo todas las particiones, guardarla en la tarjeta *SD*.
    a. ***backup***: Toma una copia de seguridad *Nandroid*

b. ***restore***: Restaura una copia de seguridad previamente tomada. Entrando en esta opción se presenta una lista de las copias de seguridad existentes de la tarjeta *SD* que se pueden elegir para el proceso.

c. ***advanced restore*****:** Opción similar a la opción **6.b**, pero permite además elegir qué partes del sistema restaurar. Puede elegir restaurar el arranque, sistema, datos, caché o particiones *sd-ext.*

7. ***mounts and storage***: Permite realizar tareas de mantenimiento en las particiones internas y externas del dispositivo Android.
    a. ***mount/unmount </cache | /data | /sdcard | /system | /sd-ext>*****:** Monta o desmonta las particiones correspondientes. La mayoría de los usuarios no necesitan interactuar con estas opciones.
    b. ***format </cache | /data | /sdcard | /system | /sd-ext>***: Formatea cualquiera de estas particiones. Dicha opción es bastante delicada y en caso de utilización inadecuada se perderán los datos, especialmente del arranque y particiones del sistema. Formatear la partición del sistema eliminará la *ROM* dejando al terminal sin sistema operativo y posiblemente *brickeado* sin solución.
    c. **mount USB storage**: Activa el modo de almacenamiento masivo USB haciendo posible transferir información al terminal sin salir del modo de recuperación.
8. ***advanced*** (avanzado): Posee opciones útiles como la limpieza de la caché *Dalvik*, acción requerida antes de la instalación de la mayoría de las *ROMs*.
    a. ***reboot recovery***: Reinicia el dispositivo en modo de recuperación.
    b. ***wipe dalvik cache*****:** Borrar la caché de la máquina virtual *Dalvik*.
    c. **wipe battery stats:** Limpia las estadísticas de la batería guardados. Útil en varios escenarios cuando Android no está mostrando niveles correctos de la batería.
    d. ***report error***: En caso de errores, esta característica se puede utilizar para guardar un registro de las operaciones recientes de recuperación ClockworkMod en la tarjeta *SD* para posteriormente informar desde Android mediante el Administrador de ROMs.
    e. ***key test***: Permite realizar pruebas sobre los botones de hardware para ver si están funcionando correctamente, y ver sus códigos.

Anexo 4: Opciones de la herramienta TWRP

1. ***Install*:** La opción permite instalar *ROMs*, núcleo u otros módulos en el sistema. Algo novedoso es que soporta instalaciones múltiples, es decir, se pueden encolar componentes que serán instalados en el orden especificado.
2. ***Wipe*:** Se utiliza para limpiar las particiones que lo permiten, tales como: *System*, *Data*, *Cache*, *Dalvik Cache*, Almacenamiento Interno o Externo en caso de existir. Limpiar las particiones es generalmente una precondición para el proceso de instalación de una *ROM*, de aquí su importancia.
3. ***Backup*:** Esta opción de gran importancia permite crear una copia de seguridad *Nandroid* de la igual forma que el CWM.
4. ***Restore*:** Restaura una copia de seguridad realizada previamente.
5. ***Mount*:** Monta particiones para su uso en la instalación de algunos módulos que lo requieren.
6. ***Settings*:** Contiene las configuraciones del subsistema de recuperación.
    a. ***Zip file signature verification***: Habilita sólo la instalación de archivos .zip firmados.
    b. ***Use rm -rf instead of formatting***: Habilita la utilización de *rm –rf* para borrar información en lugar de formatear.
    c. ***Skip md5 generation during backup*:** Deshabilita la creación de la suma de verificación durante el proceso de copia de seguridad.
    d. ***Enable md5 verification of backup files***: Habilita la creación de la suma de verificación durante el proceso de copia de seguridad.
    e. ***Use military time***: Usa el formato de horario militar, seguido de *Military/Army*.
    f. ***Simulate actions for theme testing*:** Habilita la modificación de acciones durante las pruebas de tema.
    g. ***Time Zone***: Permite seleccionar la zona horaria.
    h. ***Screen***: Habilita/Deshabilita/Selecciona el tiempo de apagado de la pantalla.
    i. ***Restore defaults*:** Reinicia las configuraciones al valor por defecto.
    f. ***partition SD Card***: Crea particiones en la tarjeta *SD* para su uso con *ROMs* que permitan ***data2ext***.
    g. ***fix permissions***: Corrige los permisos de archivo en las particiones de memoria interna por defecto. Muy útil como una solución para varios errores y cierres forzosos que comienzan a aparecer a raíz de la asignación errónea de permisos sobre archivos importantes del sistema.

Anexo 5: Opciones de Android Kitchen

1. ***Set up working folder from ROM***: Crea el espacio de trabajo y desempaqueta la *ROM*. Sólo puede cocinarse una *ROM* a la vez pero permite administrar varios espacios de trabajo simultáneamente creando copias de los mismos.
2. ***Add root permissions***: Asigna permisos de superusuario.
3. ***Add* BusyBox**: Añade BusyBox (conjunto de herramientas habituales en UNIX que no están disponibles en Android).
4. ***Disable boot screen sounds***: Elimina el sonido al encender el móvil.
5. ***Zipalign all *.apk file to optimize RAM usage***: Alinea todos los paquetes *.apk* para optimizar el uso de la *RAM*.
6. ***Change wipe status of ROM***: Cambia el estado de la bandera *wipe*.
7. ***Change name of ROM***: Cambia el nombre de compilación.
8. ***Show working folder information***: Muestra toda la información de la *ROM*.
0. ***Advanced Options***: Opciones Avanzadas
99. ***Build ROM from working folder*:** Reconstruye la *ROM* a partir de la carpeta de trabajo devolviendo un archivo *.zip flasheable*.

11. **De-odex files in your ROM**: Las ROMs oficiales incorporan los paquetes divididos en un archivo .apk y otro .odex. Con esto se acelera el arranque del dispositivo la primera vez que se inicia. Sin embargo, cuando lo que se pretende es modificarlos, es un problema porque no permiten acceder al contenido por estar divididos. Esta opción los vuelve a unir. Esta opción ocasiona, como efecto secundario, una lentitud en la primera vez que se inicia la ROM, el resto de veces iniciará rápido.

12. ***Tools for boot image (unpack/re-pack/etc.)****:* Desempaqueta/empaqueta el núcleo. Opción muy delicada que permite realizar modificaciones al núcleo del sistema.

13. ***Add /data/app functionality***: Añade el soporte para el directorio */data/app*. Las aplicaciones que sean integradas en la carpeta /*system/app* no pueden estar firmadas, por este motivo, si se desea añadir más aplicaciones lo habitual es que estén firmadas, por tanto no funcionarán. La solución es añadir este directorio e incluir dichas *apps*.

14. ***Add /etc/init.d scripts support (busybox run-parts)***: Esta opción añade soporte para el directorio /etc/init.d, su utilidad es ejecutar todos los *scripts* que se encuentren en él en cada inicio del dispositivo.

15. ***Unpack data.img***: Desempaqueta el archivo *data.img*. Este archivo contiene toda la información de usuario, es decir, datos personales, preferencias e información compartida por las aplicaciones.

16. ***Sign APK or ZIP file(s)*:** Firma un apk o un zip.

17. ***Convert update-script or updater-script***: Convierte el *script* de instalación del formato *update-script* a *updater-script*. Se hace automáticamente al construir la *ROM*.

18. ***Plugin scripts***: *Plugins* de la cocina. Es posible añadir funcionalidades a partir de las plantillas definidas en la cocina para los *plugins*, permitiendo así hacerla compatible con dispositivos no abarcados en la versión actual o incluir funcionalidades nuevas.

***BACK TO MAIN MENU*:** Volver al menú principal.

Anexo 6: Posibles mejoras mediante configuraciones al archivo build.prop

- Calidad de las fotografías y vídeos:

```
# Higest Photo and Video Quality
ro.media.enc.jpeg.quality=100
ro.media.dec.jpeg.memcap=12000000
ro.media.enc.hprof.vid.bps=12000000
```

- Streaming de audio y vídeo:

```
# Enable Stagefright helps stream Video and Music Faster
media.stagefright.enable-player=true
media.stagefright.enable-meta=true
media.stagefright.enable-scan=true
media.stagefright.enable-http=true
media.stagefright.enable-record=false
```

- Mejora de sonido:

```
#Resampling
af.resampler.quality=255
persist.af.resampler.quality=255
```

- Mejoras de navegación 3G/Wifi:

```
#3G Tweak
ro.ril.hsxpa=1
ro.ril.gprsclass=12
ro.ril.hep=1
ro.ril.enable.dtm=1
ro.ril.hsdpa.category=12
ro.ril.enable.a53=1
ro.ril.enable.a52=0
ro.ril.enable.3g.prefix=1
ro.ril.htcmaskw1.bitmask=4294967295
ro.ril.htcmaskw1=15449
ro.ril.hsupa.category=7

#Network Speed tweaks
net.tcp.buffersize.default=6144,87380,1048576,6144,87380,524288
net.tcp.buffersize.wifi=524288,1048576,2097152,524288,1048576,2097152
net.tcp.buffersize.umts=6144,87380,1048576,6144,87380,524288
net.tcp.buffersize.gprs=6144,87380,1048576,6144,87380,524288
net.tcp.buffersize.edge=6144,87380,524288,6144,16384,262144
net.tcp.buffersize.hspa=6144,87380,524288,6144,16384,262144
net.tcp.buffersize.hsdpa=6144,87380,1048576,6144,87380,1048576
net.tcp.buffersize.evdo_b=6144,87380,1048576,6144,87380,1048576
```

- Deshabilitar informes a Google:

```
#Disable Google Report service
profiler.force_disable_err_rpt=1
profiler.force_disable_ulog=1
```

- Ahorro de batería:

```
# Save Some battery
wifi.supplicant_scan_interval=500
power.saving.mode=1
pm.sleep_mode=1
```

- Rendimiento:

```
# Perfomance Tweaks
video.accelerate.hw=1
debug.performance.tuning=1
persist.sys.ui.hw=0
debug.sf.hw=1
debug.fb.rgb565=0
persist.sys.composition.type=gpu
```

```
persist.sys.use_dithering=0
persist.sys.purgeable_assets=1
# Disable usb debugging notification
persist.adb.notify=0

# Disable Bytecode Verification
dalvik.vm.checkjni=false
dalvik.vm.dexopt-data-only=1
dalvik.vm.verify-bytecode=false
dalvik.vm.lockprof.threshold=250
dalvik.vm.jmiopts=forcecopy
dalvik.vm.dexopt-flags=v=n,o=v,m=y
```

- Eliminar el retraso en el tono de llamada y aumentar la sensibilidad del sensor de proximidad:

```
#Remove Ring Delay
ro.telephony.call_ring.delay=0
mot.proximity.delay=15
ro.lge.proximity.delay=15
ro.kernel.android.checkjni=0
```

- Mejorar el comportamiento al hacer scrolling:

```
#Better scrolling
windowsmgr.max_events_per_sec=300
ro.max.fling_velocity=12000
ro.min.fling_velocity=8000
```

GLOSARIO DE TÉRMINOS

Código abierto (Open Source)

Expresión con la que se conoce al software distribuido y desarrollado libremente. Se enfoca más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto destacan en el software libre(Open Source Initiative, 2015).

Firmware

Bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria, normalmente de lectura/escritura (*ROM*, *EEPROM*[18], *flash*, etc.), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.

Framework

La palabra inglesa "*framework*" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un *framework* o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto(Riehle, 2000).

Kernel

En informática, un **núcleo** o ***kernel*** es un software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado (conocido también como modo núcleo). Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema (What is the GNU Hurd?, 2010). Como hay muchos programas y el acceso a los dispositivos es limitado, también se

[18] *EEPROM* (Memoria de sólo lectura Eléctricamente Programable y Eliminable) es un tipo de memoria no-volátil utilizada en ordenadores y otros dispositivos electrónicos para almacenar pequeñas cantidades de datos.

encarga de decidir qué programa podrá hacer uso de un dispositivo y durante cuánto tiempo, lo que se conoce como multiplexado. Acceder al hardware directamente puede ser realmente complejo, por lo que los núcleos suelen implementar una serie de abstracciones. Esto permite esconder la complejidad, proporcionando una interfaz limpia y uniforme, lo que facilita su uso al programador.

Usuario Root

En sistemas operativos del tipo Unix, *root* es el nombre convencional de la cuenta de usuario que posee todos los derechos en todos los modos (mono o multi usuario). *Root* es también llamado superusuario. Normalmente esta es la cuenta de administrador. El usuario *root* puede hacer muchas cosas que un usuario común no puede, tales como cambiar el dueño o permisos de archivos y enlazar a puertos de numeración pequeña(Linfo, 2007). En el marco de las aplicaciones y dispositivos Android, dicho usuario posee libertades para la instalación/uso de aplicaciones que lo requieren, además de otras posibilidades avanzadas. Cuando un dispositivo tiene acceso al usuario *root* se dice que esta *rooteado*.