Desarrollo de un sistema embebido móvil de bajo costo utilizando la tarjeta beaglebone black y programas de código abierto

Edgar Alejandro Rivas Araiza

Universidad Autónoma de Querétaro, Facultad de Ingeniería, Teléfono: 442-274-4637

erivas@uaq.mx

Alexander Rodríguez Rosales

Universidad Autónoma de Querétaro, Facultad de Ingeniería, Teléfono: 442-271-2588

arodriguez113@alumnos.uaq.mx

Estefanía Desiree Avalos Rivera

Universidad Autónoma de Querétaro, Facultad de Ingeniería, Teléfono: 993-300-8885

estefania.desiree.avalos.rivera@gmail.com

Resumen

El presente trabajo describe la implementación del sistema operativo libre Android, sobre una tarjeta embebida de bajo costo llamada Beaglebone Black. El objetivo es desarrollar un sistema embebido, que funcione de plataforma para probar aplicaciones móviles, interfaces gráficas, controladores de dispositivos, entre otros proyectos que se están llevando a cabo en la facultad de ingeniería. Se presenta las funcionalidades de la capa de interface de radio (*RIL*), y cómo se puede configurar en el sistema operativo. Se explica también como manejar el modem GSM Enabler III de Enfora, a través del *RIL* y la interface serial. Finalmente se presentan los resultados obtenidos, que fueron la posibilidad de hacer llamadas y enviar mensajes de texto a través del sistema, haciendo uso de los programas desarrollados en la facultad para sistemas Android.

Palabras Claves: Android, beaglebone, código abierto, ril, sistemas embebidos.

1. Introducción

En México, de acuerdo a un estudio sobre los hábitos de los usuarios de internet realizado por la AMIPCI (Asociación Mexicana de Internet) en el año 2014, se reporta que 5 de cada 10 internautas se conectan a través de su teléfono inteligente (*smartphone*), siendo el dispositivo de conexión a Internet más usado después de la laptop y la PC [1]. Debido a esta nueva tendencia tecnológica de tener acceso a Internet desde cualquier lugar, equipos como las tabletas, teléfonos inteligentes, equipos de navegación global, consolas de videojuegos portátiles, han causado un gran impacto en el mercado de consumo de equipos electrónicos [2]. Tecnologías como el servicio de datos celular 3.5G y 4G, Wifi, WiMAX, Bluetooth, NFC, y GPS son soportados por casi cualquier equipo móvil ya sea para tener acceso a la Internet, compartir ficheros, fotos, música, o incluso tener acceso a servicios empresariales que permiten a los trabajadores tener acceso a Intranets desde donde quiera que se este se encuentre [3]. La demanda de este tipo de productos ha permitido el desarrollo comercial de los mismos, así como también la investigación de nuevas tecnologías que permitan brindar novedosas soluciones, así como funcionalidades cada vez más integrales, como por ejemplo la interacción con el Internet de las cosas [4].

Debido a este crecimiento ágil de la tecnología móvil, ha crecido el interés en impulsar la línea de investigación en sistemas embebidos en la UAQ. Actualmente se trabaja en varios proyectos, entre los cuales está el desarrollo de un dispositivo móvil con fines académicos. La finalidad del mismo es crear una plataforma, sobre la cual no solo se pueda trabajar aplicaciones y funcionalidades típicas de un equipo móvil como un teléfono inteligente, o tableta, sino que también permita el aprender más sobre los sistemas embebidos, y así aplicarlos en otras áreas como las redes de sensores inalámbricos [5], edificios inteligentes [6], internet de las cosas, telemetría [7], gestión de la energía [8], por mencionar algunas.

Un aspecto importante del proyecto es el interés en utilizar programas de código abierto [9] también conocidos como *freesoftware* [10], u *opensource* [11]. La capacidad y funcionalidad de los mismos ha sido ampliamente demostrada [12], por lo que el interés tanto de parte de usuarios finales, como empresas [13], oficinas gubernamentales [14], y centros de investigación [15] ha aumentado significativamente en los últimos años [16]. El sistema operativo seleccionado como base del proyecto es el sistema Android 4.2 (*Jelly Bean)*, por varias razones, como el amplio soporte para plataformas embebidas [17], variedad de aplicaciones disponibles libremente [18], así como también la disponibilidad de herramientas para programación de forma gratuita.

En la parte de sistemas embebidos se optó por la tarjeta de desarrollo beaglebone black, la cual es conocida por su relación de precio versus funcionalidades. Las características del sistema se explicarán de forma más amplia en una sección posterior. Como referencia se aclara que este proyecto fue comenzado sobre otra plataforma conocida como AM335x Starter Kit [19] de Texas Instrument, la cual posee más periféricos que la beaglebone black, sin embargo tiene menos potencia de procesamiento, es mucho más costosa, y su diseño electrónico es complejo. Además de esto, tiene ciertas restricciones para trabajar con tarjetas externas, debido a que fue diseñada para trabajar principalmente con los periféricos que trae de fábrica. Por estas razones se decidió cambiar la plataforma por una con arquitectura abierta, bajo costo, con una comunidad de usuarios numerosa y creciente, y que tuviera un sistema electrónico más simple, por lo que en base a estos requerimientos llegamos a la plataforma actual.

El estado del arte en sistemas embebidos es diverso, debido a su portabilidad, bajo consumo, y flexibilidad en su diseño [21]. Estas características en conjunto con restricciones como la cantidad de memoria del sistema, baja capacidad de procesamiento, y sistema de alimentación por pilas, permiten el desarrollo de diversas investigaciones y desarrollos tecnológicos dirigidos a optimizar los recursos de estos dispositivos. Líneas de investigación como criptografía [20], implementan algoritmos

para procesamiento en paralelo haciendo uso de la computación en la nube, para mejorar la seguridad de los datos en los dispositivos móviles. Otro caso de estudio interesante es el de realizar una descripción del terreno utilizando algoritmos implementados en sistemas móviles [22]. Como es notable, la oportunidad de desarrollo de nuevas aplicaciones para sistemas embebidos móviles es muy amplia, por eso uno de los objetivos es implementar una plataforma sobre la cual se puedan realizar desarrollo de programas, de modo que los estudiantes puedan trabajar en proyectos que tengan el potencial de convertirse en productos finales, gracias a todas las ventajas de la ideología del conocimiento libre.

1.1. Vistazo general al sistema Android

En este proyecto se implementó el sistema operativo Android, basado en Linux y desarrollado por Google. Este ha logrado convertirse desde su lanzamiento beta en 2007 [23], en una de las principales plataformas móviles en el mercado y es cada vez más popular en los teléfonos inteligentes y tablets [24]. En términos prácticos, Android es una aplicación desarrollada sobre el núcleo (*kernel*) Linux, lo que facilita su rápido despliegue en diferentes plataformas. Además de los dispositivos móviles, Linux domina el mercado de los sistemas embebidos, sin embargo la interfaz de usuario para los desarrolladores antes de la llegada de Android era casi inexistente, y con las que se contaban eran limitadas. Esto es más que una funcionalidad técnica, ya que los desarrolladores que crean dispositivos orientados a encarar al usuario, también deben lidiar con factores de interacción humano-computadora. Aquí es donde Android, que provee un sistema operativo robusto y rápido, ayuda a los diseñadores que deben presentar al usuario una experiencia eficiente y fácil de usar, con funcionalidades como la interfaz touchscreen que brinda un fácil acceso a las aplicaciones multimedia mientras usa un mínimo de recursos [25].

La clave del éxito de Android es que su código fuente es abierto y es gratis para su uso en cualquier proyecto ya sea comercial o no. Para que los desarrolladores implementen

sus aplicaciones embebidas, Android provee un paquete completo de componentes de programación en la parte superior del núcleo Linux. Este último provee las funciones básicas de gestión del sistema para controlar procesos, periféricos, memoria y seguridad. El núcleo maneja la creación de redes y los controladores de dispositivos, lo que facilita la interfaz con el periférico. El entorno de desarrollo, que normalmente trabaja en GNU Linux, permite ejecutar código desde el dispositivo, el cual puede estar conectado vía USB [26]. Vale aclarar que al hablar de Linux [27], se trata del núcleo básico para sistemas operativos como lo es en el caso de Android. GNU Linux por otra parte, es un sistema operativo de código abierto, el cual al igual que Android, usa a Linux como núcleo base.

En los últimos años diferentes versiones de Android OS han sido liberadas. Al inicio del proyecto de la plataforma móvil, se trabajó con la versión Ice Cream Sandwich 4.0.3, sin embargo se avanzó hasta la distribución Android JellyBean 4.2.2, o *API* nivel 17. Los niveles *API* (Interfaz de programación de aplicaciones) describen como es la comunicación entre las funciones del dispositivo y su funcionalidad, y se incrementan conforme a su funcionalidad. En los niveles del 3 al 10, Android fue hecho específicamente para teléfonos inteligentes, y del 11 al 13 fueron hechos solo para tablets. A partir del nivel 14, que se inicia con Ice Cream, se unieron las funcionalidades para ambos teléfonos y tablets [28].

Existen varios proyectos de portar el soporte de Android a las diferentes plataformas embebidas. El que se seleccionó para el presente proyecto es el Rowboat de Texas Instrument [29], primeramente por tener el código fuente completo, y por soportar la compatibilidad con el procesador y periféricos de la tarjeta beaglebone black. Además estas fuentes han sido utilizadas desde el inicio del proyecto, por lo que se tiene documentación interna relevante para procesos como la compilación, edición del código, configuración del entorno de programación, y otros detalles.

1.2. Características de la BeagleBone Black

La tarjeta BeagleBone Black (BBB) es una plataforma embebida de bajo costo (al momento de escribir el documento tiene un costo de USD 45.00 en adelante), y está diseñada para probar aplicaciones propias de manera rápida y fácil directamente en el dispositivo. Es producida por *CirtcuitCo* para la organización sin fines de lucro BeagleBoard.org, y fue producida por primera vez en 2013. Es una computadora con un solo procesador de la línea ARM Cortex A8 @ 1GHz. La BBB es compacta, posee soporte para diversas interfaces de comunicación como I2C, SPI, UART, HDMI, USB, Ethernet, JTAG, posee, 512MB de memoria RAM, tiene soporte para almacenamiento de memoria externo por microSD, además de un acelerador gráfico 3D. La descripción completa de las características de la BBB se encuentra en la tabla 1.

	Feature	
Processor	Sitara AM3358BZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 800MHZ	
Onboard Flash	4GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 ,1920x1080@24Hz w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	

Tabla 1. Especificaciones de la BBB [30].

La BBB tiene soporte para varios sistemas operativos, entre ellos Android, Linux (Debian, Ubuntu, Arch), FreeBSD, entre otros [30]. Una de las características claves de esta tarjeta es que ha sido diseñada con la filosofía de electrónica abierta o libre (*open hardware)* que es la contraparte de código abierto pero implementado en diseño electrónico. Esto quiere decir que desde la lista de partes, el diseño esquemático completo, ficheros para maquinar el circuito, y demás detalles están disponibles sin restricción en Internet, desde la página de beagleboard.org [31]. Esto ha dado pie a que diversas empresas diseñen tarjetas externas compatibles con la tarjeta, llamadas *cape* las cuales se conectan directamente a la tarjeta por medio de conectores diseñados para tal fin (vease fig. 1). Entre esos *cape* se encuentra también pantallas LCD, que se pueden conectar directamente, para formar un dispositivo móvil de forma fácil y segura. Una de las pantallas soportadas es la CS-BBB-EXP50C, de la cual se hablará en la siguiente sección.

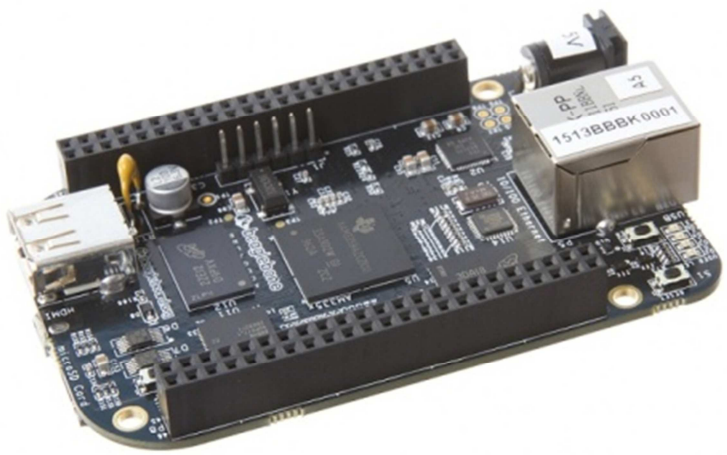

Fig. 1. Tarjeta BeagleBone Black [30].

1.3. Pantalla capacitiva CS-BBB-EXP50C

La pantalla seleccionada para el proyecto es la CS-BBB-EXP50C fabricada por Chipsee. Esta tiene un tamaño de 5”, una resolución de 800x480 pixeles, soporte multitáctil para 5 puntos, trae 5 botones de pulsasión de uso general, y trae una interface RS232 (véase fig. 2). Tiene soporte para sistemas Android, GNU Linux, y

Windows CE [32]. La pantalla capacitiva tiene un rango de sensibilidad mayor a la resistiva, por lo que permite una mejor experiencia para el usuario al momento de utilizarla. El proceso de compilación de los controladores se describe en una sección posterior.

1.4. Capa de Interface de Radio (RIL)

Dentro de la arquitectura de Android (véase fig. 3) [33], se cuenta con aplicaciones para realizar llamadas, mandar SMS, manejo de Wifi, y demás funciones básicas con las que cuenta cualquier teléfono.

Fig. 2. Pantalla LCD capacitiva multitáctil.

El compendio de servicios de telefonía, es inicializado y puesto en marcha durante el inicio del sistema y todas las peticiones realizadas desde las aplicaciones a través del *API* serán redirigidas al *RIL* por dichos servicios. El *RIL* (Radio Interface Layer) es el puente entre los servicios de telefonía de Android y los dispositivos e incluye soporte para radios basados en *GSM* (sistema global para la comunicación móvil). El RIL consiste en 2 componentes primarios: el *RIL daemon*, y el *vendor RIL.*

RIL Daemon (*rild*): Arranca durante el inicio de sistema de Android y viene incluido en el sistema en "hardware/ril/" como "rild", y trae una referencia para implementación

"reference-ril". Procesa toda comunicación desde los servicios de telefonía Android y despacha llamadas al *vendor* como comandos solicitados. RILD se encarga de encontrar la librería adecuada para inicializar el vendor RIL y desplegar todas las funciones de Vendor a la capa superior.

Vendor RIL (*ril.h*): es una librería específica para cada modem, la cual procesa toda la comunicación con el dispositivo de radio y despacha llamadas al *rild* mediante comandos no solicitados.

La librería *ril.h,* define los estados y variables del *RIL* mediante funciones. Para los comandos solicitados del *RIL,* el *vendor* debe proveer las funciones para manejarlos, se encuentran definidos con el prefijo RIL_REQUEST.

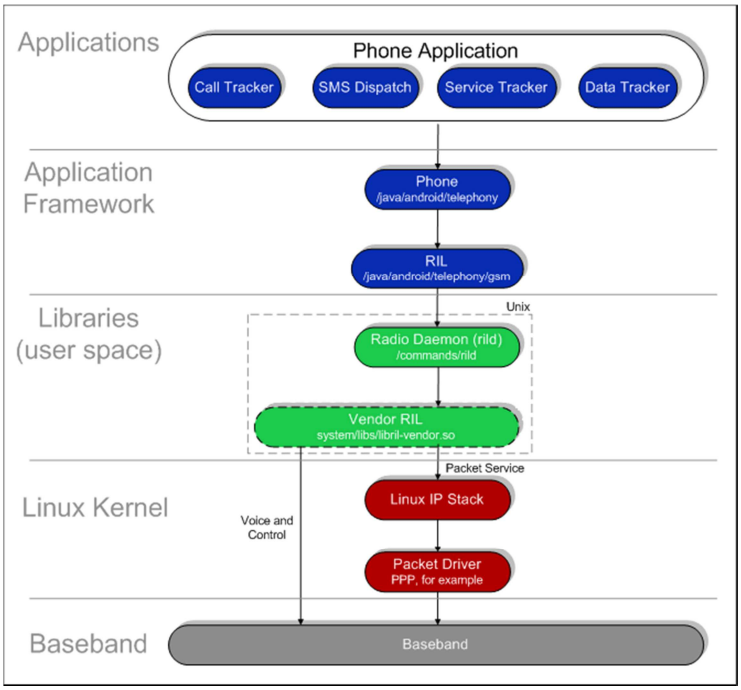

Fig. 3. Arquitectura de comunicación de Android e interacción con el RIL.

El *vendor RIL* usa funciones de respuesta para invocar comandos no solicitados. El *RIL* se maneja por comandos *AT*, que son instrucciones codificadas enviados desde el equipo que conforman un lenguaje de comunicación entre el hombre y el modem.

El interés de trabajar con el RIL en el presente proyecto es el de implementar una comunicación por GSM para realizar llamadas y envío de mensajes de texto. Para tal fin se empleó el módulo ENFORA GSM0308-11 Enabler III, el cual soporta también comunicación de datos por GPRS, sin embargo esta última funcionalidad estará implementada en la siguiente fase del proyecto. [24]

2. Desarrollo

A continuación se presenta la información sobre el desarrollo del sistema propuesto.

2.1. Entorno de compilación

Para llevar a cabo las tareas de compilación, se procede a configurar el entorno tal como se describe en [34]. En este proyecto se trabajó en un servidor de compilación con el sistema operativo CentOS 7, y una máquina virtual con Ubuntu 12.04 de 64 bit. Para la virtualización se utilizó la herramienta *docker* [35], la cual permite tener máquinas virtuales mínimas llamadas contenedores (*containers*), quienes utilizan parte del núcleo del sistema operativo anfitrión para trabajar de forma similar a una aplicación nativa. El *docker* posee muchas ventajas por sobre otros virtualizadores, como por ejemplo el uso de *dockerfiles*, que permite compartir la configuración de una máquina virtual, para que sea clonada en cualquier otro equipo, bajando la información desde el Internet. Un ejemplo de *dockerfile* se muestra a continuación, donde se puede apreciar la configuración del entorno de compilación de forma automática, la creación de usuarios, contraseñas, selección de la distribución y versión del sistema operativo, entre otros detalles:

```
FROM ubuntu:precise-20150320
MAINTAINER Alexander Rodriguez R <arodriguez113@alumnos.uaq.mx>
# Setup for Java
RUN echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" \
        >> /etc/apt/sources.list.d/webupd8.list && \
    apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886 && \
    echo oracle-java6-installer shared/accepted-oracle-license-v1-1 select true | debconf-set-
selections

# /bin/sh points to Dash by default, reconfigure to use bash until Android
# build becomes POSIX compliant
RUN echo "dash dash/sh boolean false" | debconf-set-selections && \
```

```
    dpkg-reconfigure -p critical dash

# Keep the dependency list as short as reasonable
RUN apt-get update && \
    apt-get install -y bc bison bsdmainutils build-essential curl \
        flex g++-multilib gcc-multilib git gnupg gperf lib32ncurses5-dev \
        lib32readline-gplv2-dev lib32z1-dev libesd0-dev libncurses5-dev \
        libsdl1.2-dev libwxgtk2.8-dev libxml2-utils lzop \
        oracle-java6-installer oracle-java6-set-default \
        pngcrush schedtool xsltproc zip zlib1g-dev \
               uboot-mkimage tofrodos sudo vim && \
    apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

ADD https://commondatastorage.googleapis.com/git-repo-downloads/repo /usr/local/bin/
RUN chmod 755 /usr/local/bin/*
# RUN cpan App::cpanminus && cpanm Switch

# The persistent data will be in these two directories, everything else is
# considered to be ephemeral
VOLUME ["/tmp/ccache", "/home/usuario"]

#
# user account
#
ENV MYNAME usuario
ENV MYPASS contraseña
ENV HOME /home/$MYNAME

RUN echo "User: $MYNAME Pass: $MYPASS"
RUN useradd --create-home -d $HOME --shell /bin/bash --user-group --groups adm,sudo $MYNAME
RUN echo "$MYNAME:$MYPASS" | chpasswd

USER $MYNAME
WORKDIR $HOME

# Improve rebuild performance by enabling compiler cache
ENV USE_CCACHE 1
ENV CCACHE_DIR /tmp/ccache
```

Se recomienda trabajar con *docker* si se trabaja en diferentes proyectos al mismo tiempo, con un grupo de desarrolladores, y que necesitan diferentes entornos de compilación. En el caso presente, fue una herramienta muy útil y versátil para poder aislar los entornos de los diferentes sistemas operativos, además de la facilidad con que se puede compartir, o reproducir los entornos de trabajo, logrando una respuesta aceptable por parte del servidor ante la demanda de servicio.

2.2. Habilitación del soporte para la LCD capacitiva

Una vez que se cree el entorno de compilación se puede trabajar con las fuentes del *Rowboat*, y del Android distribuido con la pantalla LCD, para poder habilitarle el soporte. Como ya mencionamos anteriormente, la pantalla necesita de controladores tanto para la pantalla LCD como para la parte táctil. Las versiones de Android en ambos casos era

exactamente la misma, Android JB 4.2.2 DevKit 4.1.1, permitiendo que la habilitación fuera casi transparente entre ambos repositorios.

El primer problema a vencer fue que a pesar de que ambas distribuciones eran iguales, la versión del fabricante del LCD resultó ser una versión minimalista del JB, quiere decir que el repositorio no incluía todo el código fuente. De esta manera se volvió más difícil encontrar cuales podrían ser las diferencias entre ambos, por lo que se tuvo que idear una forma de poder encontrar cuales eran los ficheros que se encontraban en el repositorio del JB mínimo, y que no estaban en el *Rowboat*, además de eso cuales ficheros existían en ambos repositorios pero no eran iguales. En esta fase del proyecto no se trabajó con *git*, que es una herramienta de control de versiones que contiene diferentes herramientas para realizar dicha tarea, debido a la complejidad que añadía el manejo de dicha herramienta pues aunque es útil, no es tan intuitiva para el usuario. De este modo lo que se hizo fue desarrollar una serie de *scripts* en *Bash* utilizando la herramienta *diff* como reemplazo del *git*. El primer *script* permite conocer cuales ficheros están en el JB mínimo y no en el *Rowboat*, además permite saber que ficheros son diferentes entre repositorios. El código del script se muestra a continuación:

```
#!/bin/sh
dir1=$1
dir2=$2
IFS=$'\n'
#for file in $(grep -Ilsr -m 1 '.' "$dir2"); do
#   diff -q "${file/${dir1}/${dir2}}" "$file"
#done
for file in $(grep -Ilsr -m 1 '.' "$dir1"); do
   diff -q "$file" "${file/${dir1}/${dir2}}"
done
```

Para correr este *script* se puede realizar de la siguiente manera:

```
script.sh jb-minimo/ rowboat/ > ficheros_diff.txt 2> ficheros_ausentes.txt
```

En donde *jb-minimo* y *rowboat* son las carpetas de cada repositorio y *script.sh* es el nombre del *script.* Como resultado se obtendrán dos archivos de texto, *ficheros_diff.txt* que es una lista de ficheros que están en ambos repositorios pero no son iguales, y en *ficheros_ausentes.txt* están los que aparecen en *jb-minimo* pero no en *rowboat*. Este

último arrojó un listado en donde se reconocieron todos los ficheros correspondientes al controlador tanto de la pantalla LCD como del multitáctil, por lo que solo se necesita copiarlos al repositorio del *Rowboat*. Luego de filtrar adecuadamente el listado de ficheros diferentes, se puede pasar a utilizar otro *script* el cual creará un grupo de parches, uno por cada fichero, que luego aplicaremos al repositorio del *Rowboat*. Esto permitirá que los ficheros de compilación incluyan los controladores de la pantalla, al momento de empezar con el proceso. El script se presenta a continuación:

```
#!/bin/sh
LINEAS=`wc -l $1 | cut -d" " -f1`
for i in $( seq 1 $LINEAS );
       do
       FICHERO=`awk NR==$i $1`
       diff -u $FICHERO > parche$i.patch
done
```

Una vez ya se tengan los parches listos, se pueden aplicar utilizando el siguiente *script*:

```
#!/bin/sh
LINEAS=`wc -l $1 | cut -d" " -f1`
for i in $( seq 1 $LINEAS );
       do
       FICHERO=`awk NR==$i $1|cut -f1`
       patch $FICHERO < parche$i.patch && echo "parche $i listo"
       #echo $FICHERO
done
```

Una vez se apliquen los parches, se procede al proceso de compilación tal como se describe en [34], se crea la imagen en la memoria microSD y se procede a activar los interruptores necesario en el cape del LCD. Luego del arranque de la BBB, esta se conecta a una estación de trabajo mediante la interface USB. En el dispositivo activamos el modo de depuración para USB (*USB Debugging mode)* en ajustes, para que se pueda realizar cualquier tarea de mantenimiento o depuración en el dispositivo.

2.3. Configuración del *RIL*

Para activar la configuración del *RIL*, se utilizó información explicada en [34] y también se añadieron algunos puntos específicos para el modem de Enfora. Este modem tiene una distribución de *vendor RIL* de código abierto [36], sin embargo está diseñada para las primeras versiones de Android, por lo que existen problemas en la implementación

directa en el código fuente. La solución fue adaptar el *reference ril*, que es una distribución de ejemplo que viene con el código fuente del *Rowboat,* para que cumpliera con los requerimientos de comunicación del modem.

Tal como se explicó anteriormente, la función del *RIL* es gestionar el control de la comunicación entre diferentes periféricos y las aplicaciones del Android, en este caso del modem GSM. Esto se logra a través del envío de comandos *AT*, por lo que se debe consultar el manual de comandos AT soportados por el modem [37], para poder adaptar el *reference ril* para que lo controle adecuadamente. Esta parte es esencial para el correcto funcionamiento de llamadas y envíos de mensajes, pues aunque la aplicación de llamadas funcione, si no puede comunicarse con el modem adecuadamente a través del *RIL*, estas tareas no se cumplirán. Incluso puede caer en un ciclo infinito en que el sistema intenta establecer la comunicación reiniciando el *rild*, más sin embargo esto no causará más que un alza del consumo energético de la tarjeta por un uso continuo e innecesario del procesador.

Referente a la configuración del *Rowboat*, es necesario habilitar la funcionalidad de telefonía, deshabilitar inicialmente lo relacionado con *APN* si no deseas activar el servicio de GPRS, y aplicar estos cambios en diversos ficheros. Los más importantes se presentan a continuación:

- `build/target/product/full_base.mk`
- `build/target/product/full_base_telephony.mk`
- `build/target/product/generic_no_telephony.mk`
- `build/target/product/telephony.mk`
- `kernel/arch/arm/mach-omap2/board-am335xevm.c`
- `system/core/init/property_service.c`
- `system/core/rootdir/ueventd.rc`
- `device/ti/am335xevm/device.mk`
- `device/ti/am335xevm/am335xevm.mk`
- `device/ti/beagleboneblack/overlay/frameworks/base/core/res/res/values/config.xml`
- `device/ti/beagleboneblack/init.am335xevm.rc`

3. Resultados

Terminada esta fase del proyecto, obtuvimos un sistema embebido móvil completamente funcional, desde donde se podían hacer llamadas, enviar SMS, instalar aplicaciones de terceros, con una pantalla LCD capacitiva multitáctil, con opción para reproducción de video y navegación por Internet. La conexión entre el dispositivo y el modem se logró a través de un convertidor USB-serial, y se utilizó la conexión a auriculares y micrófono del propio módulo del modem para hacer las llamadas, debido a que la BBB no trae internamente salida o entrada de audio. En la fig. 4 se puede observar el sistema interconectado con el modem, y la aplicación de telefonía trabajando.

El sistema se ha logrado desarrollar completamente con programas de código abierto, desde el entorno de compilación, el sistema operativo, incluso la electrónica es abierta, minimizando gastos, y explotando las capacidades que nos permite tener acceso al código fuente de proyectos complejos lo es el sistema operativo Android.

Fig. 4. Muestra del sistema embebido móvil en funcionamiento.

4. Discusión

Entre los puntos importantes que se lograron con el sistema obtenido está el obtener una plataforma más robusta en cuanto a potencia de procesamiento; se mantuvo un consumo bajo de energía dado que la tarjeta tiene un mínimo de periféricos incluidos; se obtuvo un dispositivo móvil que incluye una pantalla LCD de 5” que puede ser utilizada para cualquier utilidad como un teléfono inteligente, un equipo de medición en campo, sistema de posicionamiento, o cualquier otro tipo de uso; además se logró tener acceso a la red celular por medio de un modem GSM.

5. Conclusiones y trabajos futuros

Queda demostrado que se puede desarrollar sistemas embebidos con tecnología actual, haciendo uso de las herramientas disponibles de código y electrónica abierta. No es obligatorio ya tener que pagar por costosas licencias, ni quedarse solo con las funcionalidades que los programas traen por defecto. Hoy día se puede no solo descargar aplicaciones de libre distribución, sino que también se tiene acceso a sus fuentes, para poderlo modificar según sea la necesidad.

Se puede utilizar un modem GSM de bajo costo en conjunto con un sistema embebido para tener acceso a la red celular, ya sea para realizar llamadas, enviar mensajes de texto, o también para comunicarse por la red de datos móvil. La diversidad de interfaces que tiene la BBB permite manejar múltiples periféricos simultáneamente, convirtiéndola en una solución adecuada para diversas aplicaciones.

Como trabajos a futuro está la habilitación del soporte para GPRS a través del modem Enfora, diseñar y construir un *cape* propio para manejar las entradas y salidas de audio y para montar el modem de forma directa a una placa electrónica. También está el implementar *git* para gestionar el control de versiones del proyecto, diseñar y crear una carcasa para el dispositivo móvil, y como un producto derivado está el desarrollar de

Pistas Educativas Año XXXVI - ISSN 1405-1249
Certificado de Licitud de Título 6216; Certificado de Licitud de Contenido 4777; Expediente de Reserva 6 98 92

cero una plataforma propia utilizando el procesador ARM Cortex A8, para un uso determinado según se requiera.

Finalmente este ejercicio arroja como resultado obtener una plataforma lista para aplicaciones embebidas, utilizando Android como sistema operativo base, lo que abre las puertas a futuras implementaciones mucho más complejas, en donde se podrá implementar soluciones propias haciendo uso de las funcionalidades que este sistema operativo brinda a los desarrolladores, todo a través del código abierto y la liberación del conocimiento.

6. Referencias

[1] Estudio sobre los hábitos de los usuarios de Internet en México 2014. https://www.amipci.org.mx/estudios/habitos_de_internet/Estudio_Habitos_del_Internauta_Mexicano_2014_V_MD.pdf. Acceso: 6 de junio de 2015.

[2] Smartphones en México: más marcas, más modelos, más competencia. http://www.elfinanciero.com.mx/empresas/marcas-y-modelos-de-smartphones-multiplican-su-presencia-en-mexico.html. Acceso: 5 de junio de 2015.

[3] J. Grover, "Android forensics: Automated data collection and reporting from a mobile device". J. Digital Investigation. Vol. 10. Suplemento. Agosto 2013. S12–S20 pp.

[4] F. Salim, U. Haque, "Urban computing in the wild: A survey on large scale participation and citizen engagement with ubiquitous computing, cyber physical systems, and Internet of Things". Int. J. Human-Computer Studies. Vol. 81. Septiembre 2015. 31–48 pp.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey". J. Computer Networks. Vol. 38. 2002. 393-422 pp.

[6] P. De Mil, T. Allemeersch, I. Moerman, P. Demeester and W. De Kimpe, "A scalable low-power WSAN solution for large-scale building automation". IEEE International Conference on Communications. Mayo 19-23, 2008. 3130-3135 pp.

[7] G. Bruzzone, M. Caccia, G. Ravera, A. Bertone, "Standard Linux for embedded real-time robotics and manufacturing control systems". J. Robotics and Computer-Integrated Manufacturing. Vol. 25. No. 1. Febrero 2009. 178–190 pp.

[8] F. Shrouf, G. Miragliotta, "Energy management based on Internet of Things: practices and framework for adoption in production management". J. of Cleaner Production. Vol. 100. 2015. 235–246 pp.

[9] M. D. Gallego, S. Bueno, F. J. Racero, J. Noyes, "Open source software: The effects of training on acceptance". Computers in Human Behavior. Vol. 49. 2015. 390–399 pp.

[10] ¿Qué es el software libre? http://www.gnu.org/philosophy/free-sw.html. Acceso: 5 de Junio de 2015.

[11] The open source definition. http://opensource.org/definition. Acceso: 5 de junio de 2015.

[12] M. Sarrab, O. M. Hussain Rehman, "Empirical study of open source software selection for adoption, based on software quality characteristics". Advances in Engineering Software. Vol. 69. Marzo 2014. 1–11 pp.

[13] Ø. Hauge, C. Ayala, R. Conradi, "Adoption of open source software in software-intensive organizations – A systematic literature review". Information and Software Technology. Vol. 52. 2010. 1133-1154 pp.

[14] A. van Loon, D. Toshkov, "Adopting open source software in public administration: The importance of boundary spanners and political commitment". Government Information Quarterly. Vol. 32. 2015. 207-215 pp.

[15] J. S. Horsburgh, S. L. Reeder, A. Spackman Jones, J. Meline, "Open source software for visualization and quality control of continuous hydrologic and water quality sensor data". Environmental Modelling & Software. Vol. 70. 2015. 32-44 pp.

[16] Y. Kuwataa, K. Takedab, H. Miura, "A Study on Maturity Model of Open Source Software Community to Estimate the Quality of Products". Procedia Computer Science. Vol. 35. 2014. 1711-1717 pp.

[17] Android 5.0 compatibility definition. http://static.googleusercontent.com/media/source.android.com/es//compatibility/android-cdd.pdf. Acceso: 5 de Junio de 2015.

[18] Distribution of free and paid Android apps in the Google Play Store from 2009 to 2015. http://www.statista.com/statistics/266211/distribution-of-free-and-paid-android-apps/. Acceso: 5 de Junio de 2015.

[19] AM335x Starter Kit. http://www.ti.com/tool/tmdssk3358. Acceso: 5 de Junio de 2015.

[20] S. Tang, B. Lv, G. Chen, Z. Peng, A. Diene, X. Chen, "Efficient hardware implementation of PMI+ for low-resource devices in mobile cloud computing". Future Generation Computer Systems, Special Section: Cloud Computing: Security, Privacy and Practice. Vol. 52. 2015. 116-124 pp.

[21] L. Oneto, A. Ghio, S. Ridella, D. Anguita, "Learning Resource-Aware Classifiers for Mobile Devices: From Regularization to Energy Efficiency". Neurocomputing, Industrial Data Processing and Analysis, 11th World Congress on Intelligent Control and Automation. Vol. 169. 2015. 225-235 pp.

[22] J. P. Suárez, A. Trujillo, J. M. Santana, M. de la Calle, D. Gómez-Deck, "An efficient terrain Level of Detail implementation for mobile devices and

performance study". Computers, Environment and Urban Systems. Vol. 52. 2015. 21-33 pp.

[23] A. Henderson, A. Prakash, Android for the beaglebone black. 1ra Ed. 2015. Packt Publishing Ltd. Birmingham, UK. 7-19 pp.

[24] Enabler IIIG Quad-Band OEM Cellular Modem. http://www.nvtl.com/files/1713/6439/6537/Enabler_IIIG_Quad_reduced.pdf. Acceso: 5 de Junio de 2015.

[25] K. Yaghmour, Embedded Android. 1ra Ed. 2013. O'Reilly Media, Inc. United States of America. 1-22 pp.

[26] C. Walls, Embedded Software. 2da Ed. 2012. Elsevier. UK. 337–363 pp.

[27] What is Linux?. https://www.kernel.org/linux.html. Acceso: 5 de Junio de 2015.

[28] Codenames, Tags, and Build Numbers. http://source.android.com/source/build-numbers.html. Acceso: 5 de Junio de 2015.

[29] Enables Android for Texas Instruments devices. https://code.google.com/p/rowboat/. Acceso: 5 de Junio de 2015.

[30] Beagleboard:BeagleBoneBlack. http://elinux.org/Beagleboard:BeagleBoneBlack Acceso: 5 de Junio de 2015.

[31] BeagleBoard Hardware Design. http://beagleboard.org/hardware/design. Acceso: 5 de Junio de 2015.

[32] CS-BBB-EXP50C. http://www.chipsee.com/cs-bbb-exp50c.html. Acceso: 5 de Junio de 2015.

[33] Radio Layer Interface. http://www.kandroid.org/online-pdk/guide/telephony.html. Acceso: 5 de Junio de 2015.

[34] J.M. Ramos-Arreguín, P. Richter, A. Takacs, M. Toledano-Ayala, E.A. Rivas Araiza, J.C. Moya- Morales, J.E. Rivas-Araiza, C.O. Mendoza-Herbert, C. Vargas-Cabrera, "Adaptación del SO Android en un AM335x para usar un módulo GSM". XI Congreso Internacional sobre Innovación y Desarrollo Tecnológico, CIINDET. 25-27 de Marzo 2015.

[35] What is docker? https://www.docker.com/whatisdocker/. Acceso: 5 de Junio de 2015.

[36] Enfora EDG308 RIL Plugin. http://omapzoom.org/?p=platform/hardware/enfora/edg308-ril.git;a=summary. Acceso: 5 de Junio de 2015.

[37] Enfora Enabler III GSM/GPRS/EDGE Radio Modem AT Command Set Reference. http://www.farnell.com/datasheets/876256.pdf. Acceso: 5 de Junio de 2015.

7. Autores

Dr. Edgar Alejandro Rivas Araiza es Ingeniero en Automatización y Control de Procesos, M. en C. en Control Automático y Dr. en Ingeniería por la Universidad Autónoma de Querétaro. Es investigador y docente de la Facultad de Ingeniería desde el 2005. De 2008 al 2011 fue coordinador del Centro de Diseño e Innovación Tecnológica (CEDIT) y actualmente es coordinador de la Maestría en Instrumentación y Control en la Facultad de Ingeniería.

Ing. Alexander Rodríguez Rosales es Ingeniero en Electrónica y Telecomunicaciones por la Universidad Tecnológica de Panamá, y es actualmente estudiante de la M. C. en Instrumentación y Control con Línea Terminal en Electrónica. Es experto en sistemas operativos GNU Linux, y su línea de investigación son los sistemas embebidos.

Ing. Estefanía Desiree Avalos Rivera es Ingeniera Mecatrónica por la Universidad Politécnica del Centro, y es actualmente estudiante de la M. C. en Instrumentación y Control con Línea Terminal en Electrónica. Su línea de investigación son los sistemas embebidos.