EXTENSIÓN DE CAPACIDADES DE TRANSMISIÓN DE DATOS EN SMARTPHONES CON SISTEMA OPERATIVO ANDROID

Ramirez Malave Johnny [(1)], Sanchez Alvarez Washington [(2)]
Facultad de Ingeniería en Electricidad y Computación (FIEC)
Escuela Superior Politécnica del Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 Vía Perimetral
Apartado 09-01-5863 Guayaquil, Ecuador
joomrami@espol.edu.ec [(1)], wansanch@espol.edu.ec [(2)]

RESUMEN

El siguiente trabajo demuestra una de las principales ventajas de usar el sistema operativo Android: el poder manipularlo para explotar plataformas móviles. En nuestro proyecto habilitamos el módulo del núcleo de android necesario para utilizar un transmisor de radio y así poder recibir datos desde puntos de difícil acceso o carentes de acceso a la red celular. En este proyecto nos enfocamos en optimizar la velocidad de obtención de datos, haciendo uso de librerías y marcos de trabajo para el núcleo de Android y el núcleo de Linux. En nuestros experimentos se analizó el rendimiento de una aplicación desarrollada para el sistema operativo Android, en base al número de paquetes perdidos durante la trasmisión y la distancia a la que fueron enviados. Este trabajo permite usar, en principio, un teléfono como un nodo de una red de sensores.

***PALABRAS CLAVE**: Android, Núcleo, Modulo FTDI, Kit de Desarrollo Nativo.*

ABSTRACT

The next paper shows the advantages of using Android's Operative System: The fact that we can modify it so we can explode the mobile platforms. In our project we enabled the needed modules so we can plug and use the radio transmitter, so we can receive data from difficult access locations, or where there is none cell net access. This project focus in optimize the data collection's speed using the libraries and frameworks for the Android Kernel and Linux Kernel. In our experiments we analyzed the performance of an application developed for the Android's Operative System, based on the number of lost data and the distance at which the transmitter was located. This work allows the use of a mobile phone such as a sensor's net node.

1. INTRODUCCIÓN

El uso de tabletas y teléfonos inteligentes en la vida diaria es una realidad palpable. Hoy en día los dos más grandes Sistemas Operativos para teléfonos inteligentes son iOS y Android, pero podemos remarcar que Android no solo es usado en estos teléfonos inteligentes o tabletas, sino que también es utilizado en dispositivos como relojes, cámaras fotográficas y hasta televisores.

La disminución de costos, la facilidad, gratuidad de las herramientas para programar, y la comercialización de teléfonos, con Android como Sistema Operativo, por parte de grandes compañías de telecomunicaciones, abrió un nicho de mercado que se encuentra en constante crecimiento para los desarrolladores de software de aplicaciones sencillas.

En el presente documento vamos a ir más allá de desarrollar una simple aplicación, se muestra como extender la funcionalidad de Android para que soporte dispositivos externos. En nuestro proyecto habilitamos los puertos de comunicación del teléfono y el núcleo del sistema operativo para utilizar una radio que recibe datos extraídos de un GPS remoto.

Primeramente se define el objetivo de este trabajo, y se explica porque se decidió desarrollar el programa sobre el sistema operativo Android. Después se explica las ventajas que tiene sobre otros sistemas operativos para dispositivos móviles y como habilitar un módulo del núcleo de Android para poder comunicarnos con un dispositivo externo basado en FTDI, se listan los pasos a seguir, y se explica el diseño de la aplicación y sus diferentes versiones. Finalmente, presentaremos datos estadísticos acerca de la eficiencia de ejecutar la aplicación desarrollada sobre un teléfono, y como se puede aprovechar la facilidad que presenta el teléfono para trasladarse a puntos de difícil acceso.

2. DEFINICIÓN DEL PROBLEMA

Los equipos utilizados para tomar datos climáticos o meteorológicos, son ubicados en lugares particulares, remotos y, en la mayoría de los casos, de difícil acceso, por ejemplo en lo alto de una montaña, en la cercanía de un volcán, o en lo profundo de la selva. Teniendo en cuenta estos factores, la toma de datos en el mismo sitio donde se encuentran estos dispositivos dificulta el trabajo y conlleva demasiado tiempo, por lo que es preferible que estos dispositivos cuenten con un sistema de conexión inalámbrica de tal manera que la descarga de información sea sencilla y rápida.

Las tecnologías inalámbricas que se pueden utilizar para realizar la conexión inalámbrica en estos lugares remotos pueden tener sus limitantes: la falta de cobertura de red celular, el rango de alcance de una red WIFI y bluetooth, los costos de comunicación vía satélite, etc. A diferencia de la gran cobertura y un relativo bajo costo que una radio puede tener.

Como se mencionó inicialmente, la ubicación de estos dispositivos genera problemas debido a su difícil acceso, por lo que pensar en el uso de un computador portátil, para realizar la descarga de información, resulta algo poco probable. Es aquí donde el teléfono inteligente, con cada vez mejores capacidades de procesamiento y su facilidad de movilización, se presenta como la mejor opción para realizar la tarea de la toma de datos.

3. OBJETIVOS

Se dividen en un objetivo general donde se explica el propósito del trabajo de una manera amplia y los objetivos específicos en donde se explicara más explícitamente nuestro propósito.

3.1 OBJETIVO GENERAL

Comprobar que el teléfono inteligente con sistema operativo Android es un dispositivo con las mismas prestaciones que un computador de escritorio y que, al cumplir las características de un software de código abierto, puede ser modificado para poder conectar dispositivos pocos comunes y así aprovechar su facilidad de transportación para proyectos en los que se necesite tomar datos en lugares de difícil acceso.

3.2 OBJETIVOS ESPECÍFICOS

- Extender el núcleo de Android con la finalidad de habilitar el módulo FTDI en el teléfono y tener la capacidad de enviar y recibir datos por medio de un puerto serial.
- Desarrollar una aplicación en Android con la ayuda del NDK, y utilizar código nativo para la comunicación entre el teléfono y una radio con interfaz usb, que recibe coordenadas geográficas, las transfiere al teléfono y el teléfono grafica dichas coordenadas en un mapa.

4. ANDROID

Es un sistema operativo basado en Linux, es decir con un núcleo que puede ser modificado según las necesidades del usuario. Al estar basado en Linux no se refiere a todo el software de aplicaciones y servicios, se refiere solo al núcleo que es el software que gestiona las llamadas al sistema y las interrupciones [1], [2].

Android posee una larga comunidad de desarrolladores que incrementan la funcionalidad de los dispositivos por medio de aplicaciones escritas, en su mayoría, en una versión personalizada del lenguaje de programación java, haciendo uso de herramientas como el Android SDK [2].

La compañía ComScore, se encarga de llevar estadísticas del mundo digital, publicó en junio del 2013 sus estadísticas sobre los teléfonos inteligentes, colocando en primer lugar a Google Android como la plataforma número 1, con el 52% del mercado de sistemas operativos para móviles tal como lo muestra la tabla I [3].

	% de teléfonos		
	Febrero - 2013	Mayo - 2013	Cambio Punto
Total Teléfonos Inteligentes Subscritos	100.0%	100.0%	N/A
Android	51.7%	52.4%	0.7
Apple	38.9%	39.2%	0.3
BlackBerry	5.4%	4.8%	-0.6
Microsoft	3.2%	3.0%	-0.2
Symbian	0.5%	0.4%	-0.1

Tabla I. Mercado de Sistemas operativos móviles a nivel mundial [3]

En la tabla II podemos observar las principales ventajas que posee android sobre su competencia, como la tecnología Comunicación de Campo Cercano (NFC) que es una tecnología de comunicación inalámbrica y android aprovecha esta tecnología para el intercambio inmediato de grandes datos entre dos dispositivos,

como música y videos [4].

S.O. Móvil	Android 4.1 Jelly Bean	iOS6	Windows Phone 8
Núcleo de Sistema Operativo	Linux	OS X	Windows NT
Código Abierto	Si	No	No
Multi-tarea	Si	Limitado	Limitado
Hardware Soportado	Una amplia variedad de dispositivos	iPhone, iPad, iPod touch	Una variedad de dispositivos
Soporte Flash	Si	No	No
Soporte Java	Si	No	No
Teclado	Físico y Virtual	Virtual	Físico y Virtual
Personalización Interface de Usuario	Si, nativo o con el uso de un sin número de aplicaciones	Limitada	Limitada
Reconocimiento de Voz	Si	Si	Si
Reconocimiento de Voz fuera de línea	Si	No	No
NFC	Si	No	Si
Automatización de tareas	Si	No	Si
Almacenamiento Extraíble	Si	No	Si
Soporte para Tabletas	Si	Si	No
Base de Datos integrada	Si	Si	Si

Tabla II. Comparación entre Android, iOS y Windows Phone 8 [5]

4.1 MÁQUINA VIRTUAL DALVIK

Google seleccionó a Java como el lenguaje para el desarrollo pero creo su propia máquina virtual llamada Dalvik que es una máquina virtual de procesos que se encargan de ejecutar aplicaciones en android, cada aplicación ejecuta su propio proceso, es decir su propia instancia de máquina virtual [6].

La máquina virtual de Java utiliza el bytecode de java para ejecutar sus instrucciones estos bytecode son códigos más abstractos que el código de máquina y es el resultado de un compilador.

El objetivo fundamental del Dalvik VM es el mismo que el de cualquier máquina virtual, permitir que el código sea compilado a un código que pueda ser ejecutado en cualquier máquina, o para el caso de Android [7], [8].

4.2 KIT DE DESARROLLO DE SOFTWARE ANDROID (SDK)

Las aplicaciones en android son desarrolladas en java usando el SDK, que incluye un depurador, bibliotecas de software, un emulador de terminal basado en QEMU, documentación, código de ejemplo y tutoriales.

El IDE oficial soportado por Android es Eclipse acompañado por el complemento llamado Herramienta de Desarrollo Android (ADT). Otras herramientas de desarrollo disponibles, son el Kit de Desarrollo Nativo (NDK) para aplicaciones o extensiones en C o C++, Google App Inventor y varios marcos de trabajo inter-plataforma de aplicaciones móviles.

4.3 KIT DE DESARROLLO NATIVO ANDROID (NDK)

El NDK ayuda a implementar parte de una aplicación en lenguaje C y C++, referido como código nativo, porque es el mismo lenguaje con que está escrito el núcleo de android. Con el NDK se puede reutilizar código y en algunos casos obtener un aumento de velocidad [9].

El NDK provee un conjunto de herramientas utilizadas para generar bibliotecas de código nativo de fuentes en lenguaje C y C++. Las cabeceras y librerías nativas que pueden ser soportadas por las futuras versiones de la plataforma Android, desde la versión 2.3 en adelante.

Entre las librerías incluidas en el NDK esta la librería JNI (Java Native Interface) que sirve como un puente entre Java y Código Nativo, es decir, permite que las aplicaciones JAVA puedan incorporar código nativo escrito en lenguajes como C, C++ y Lenguaje Ensamblador, así como código escrito en el lenguaje de programación Java. Muchas de las librerías nativas de Android dependen de la funcionalidad que proporciona

JNI para cumplir su propósito, por ejemplo, las librerías de Entrada/Salida, todo este funcionamiento se muestra en la figura 1.

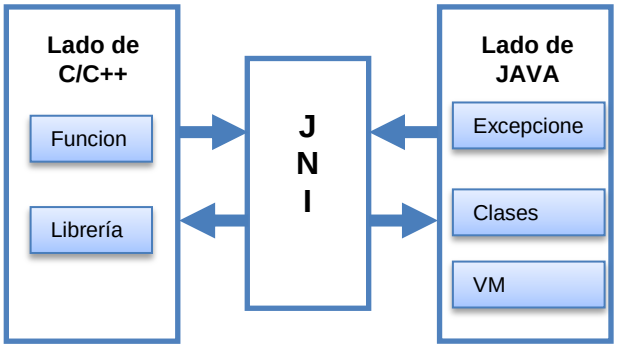

Figura 1.- JNI, el puente entre Java y Código Nativo

5. ARQUITECTURA DE ANDROID

Su arquitectura se encuentra conformada de varias capas que facilitan al desarrollador la creación de aplicaciones, esta distribución de Linux permite acceder a las capas más bajas mediante el uso de librerías diseñadas para android, para permitir que una aplicación haga uso del hardware de los teléfonos.

A la arquitectura de android se la conoce como pila, esto es porque cada una de las capas puede acceder y utilizar elementos de la capa inferior, en la figura 2 se muestra la arquitectura de android, en donde se encuentra como la capa más baja al núcleo de android, luego las librerías, después los marcos de trabajo y por ultimo las aplicaciones.

El Núcleo se encarga de administrador los diferentes recursos del teléfono, como la energía, uso de memoria, etc., y de administrar el sistema operativo en sí: procesos, elementos de comunicación (redes de trabajo), etc. Es la capa intermedia entre el hardware y las demás capas de la arquitectura [11], [12].

La capa que se encuentra sobre el núcleo la componen las librerías, que son escritas en C o C++ y son compiladas específicamente para el hardware del teléfono, la siguiente capa es la de marcos de trabajo de aplicaciones, las clases y servicios de esta capa son usadas por las aplicaciones, la mayoría de los componentes disponibles en esta capa son librerías Java que acceden a los recursos disponibles en las capas inferiores a través de la máquina virtual Dalvik [11], [12].

La última capa incluye el conjunto de todas las aplicaciones disponibles del dispositivo, ya sean si poseen o no una interface de usuario, las nativas escritas en C o C++ y las que se encargan de administrar los recursos del sistema, las que vienen instaladas por defecto las que el usuario instaló.

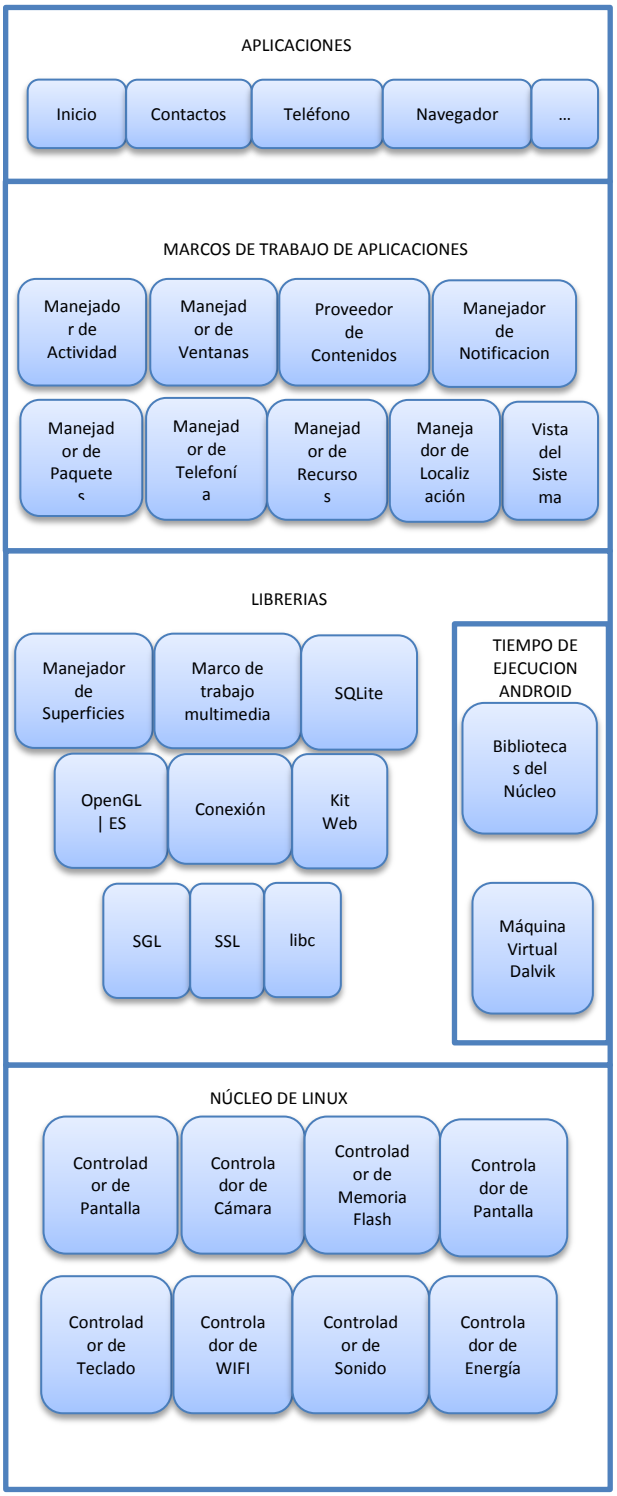

Figura 2.- Arquitectura de Android [10]

6. DISPOSITIVOS FUTUROS DE TECNOLOGÍA INTERNACIONAL (FTDI)

Los chips FTDI son chips desarrollados por la compañía escocesa de dispositivos semiconductores "Dispositivos Futuros de Tecnología Internacional", que se especializan en la tecnología USB. Los adaptadores USB utilizan estos chips FTDI para la conexión de interfaces RS232 que es un estándar para el intercambio de datos binarios [13].

Sus circuitos integrados permiten que un equipo con puerto serie o paralelo pueda manejar dispositivos con interfaz USB tanto si tiene que actuar como esclavo, recibiendo instrucciones, o maestro, es decir controlando uno o más dispositivos. El chip FTDI lo utilizan sistemas basados en microcontroladores.

La figura 3 muestra la comunicación entre el teléfono y un dispositivo externo por medio del módulo FTDI, el puerto USB del teléfono se conecta al módulo FTDI por medio de los pines USB D+ y el USB D- que son los encargados de enviar y recibir la información, luego el módulo FTDI se conecta al dispositivo externo por medio del módulo UART que es el encargado de recibir la información en formato paralelo y transformarla a formato serie y viceversa, haciendo uso de los siguientes pines, el TXD que recibe los datos, RXD que envía los datos, el RTS y el CTS que controlan el flujo de información, el RTS que solicita el envío de información y el CTS que indica que el UART está disponible para enviar información.

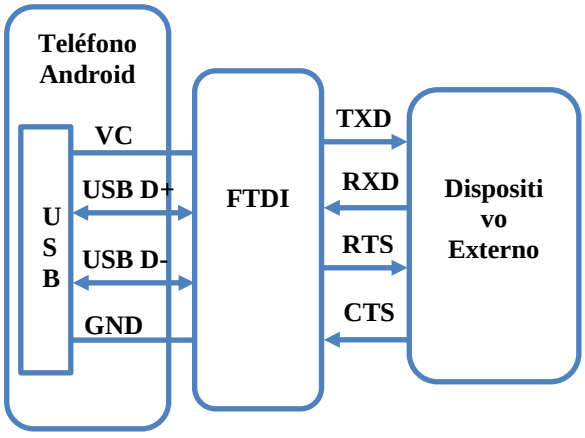

Figura 3.- Comunicación del teléfono mediante el FTDI

Soporte USB OTG (USB Anfitrión).- es una extensión del USB 2.0 que permite conectar dispositivos con el fin de usar al móvil como anfitrión que provee energía al dispositivo conectado, o un invitado, como al conectarlo a una computadora para que este funcione como un dispositivo de almacenamiento externo.

7. HABILITAR MÓDULO FTDI EN ANDROID

La figura 4 muestra los pasos que se siguió para poder habilitar el módulo FTDI en el sistema operativo android se comenzó por obtener la correcta versión del código fuente del núcleo del teléfono, luego se obtuvo permisos de superusuario, después se configuró y compiló el núcleo de android, para finalmente instalar el módulo FTDI en el teléfono.

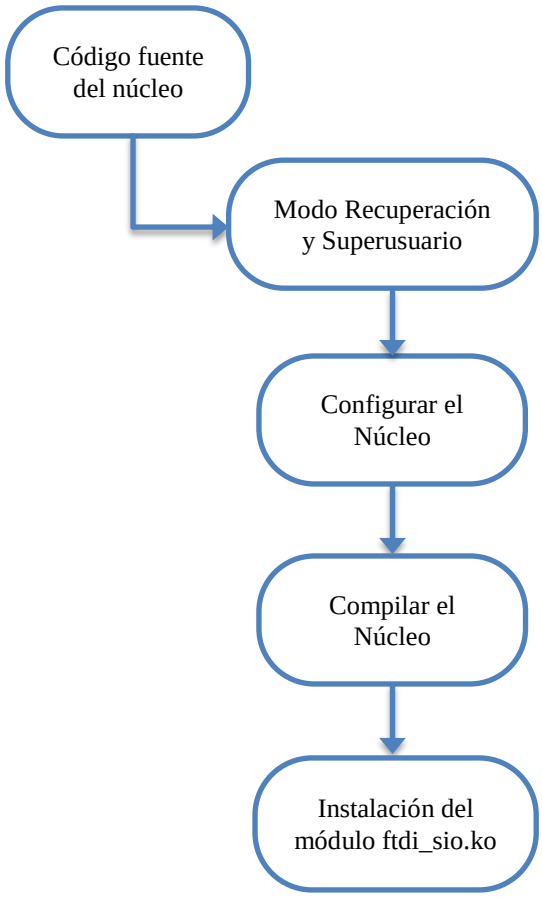

Figura 4.- Pasos para habilitar módulo FTDI

El Código fuente del núcleo es un grupo de archivos y carpetas, en este grupo de carpetas se encuentra en el directorio "Drivers", en esta carpeta están los módulos que se pueden instalar al núcleo, entre ellos el modulo que se necesitó para activar el FTDI en el núcleo, este es el archivo ftdi_sio.ko.

El superusuario, también llamado root, es el usuario que puede acceder a cualquier parte del sistema

operativo. Para realizar modificaciones o habilitar módulos en el núcleo se necesita tener privilegios de superusuario, los fabricantes de celulares mantienen bloqueado este acceso de superusuario debido a que los usuarios podrían modificar el núcleo incorrectamente por lo que se necesitan archivos adicionales para acceder como este usuario.

Para habilitar el superusuario en un teléfono inteligente con sistema operativo Android se debe ingresar al modo recuperación que es un modo donde se puede dar mantenimiento al teléfono es decir limpiar la cache, realizar copias de seguridad del sistema y permite restaurar estas copias del sistema.

La configuración del núcleo se realiza en una computadora personal y lo primero que se tiene en cuenta es tener el archivo correcto de configuración ".config" para nuestra versión de núcleo, una forma de obtener este archivo es extrayéndolo del teléfono con el comando:

$ adb pull /proc/config.gz config.gz

Algunos teléfonos, en la configuración de su núcleo no tienen habilitado la opción de mostrar el archivo "config.gz", en este caso se debe elegir un archivo de configuración de la ruta "~/kernel/arch/arm/configs/" en nuestro caso usamos el comando:

$ make ARCH=arm android_espresso_omap4430_r04_user_defconfig

Se ingresa al menú de configuración con el comando "make menuconfig ARCH=arm" y habilitamos el módulo FTDI navegando por la siguiente ruta "Controladores de Dispositivos -> Soporte USB -> Soporte Convertidor Serial -> Controlador Simple de Puerto Serial FTDI" y se coloca a la izquierda de "Controlador Simple de Puerto Serial FTDI" la opción "<M>".

Para compilar el núcleo de Android se necesita un compilador de lenguaje "C", en nuestro caso se usó el que viene integrado en el NDK, se exporta la ruta de donde se encuentra el compilador de la siguiente forma:

export PATH=/opt/android/android-ndk/toolchains/arm-linux-androideabi-4.4.3/prebuilt/linux-x86/bin/:$PATH

Para compilar el núcleo se ejecutó el siguiente comando:

$ make ARCH=arm CROSS_COMPILE=arm-linux-androideabi- -j4

Después de compilar el núcleo se transfieren los archivos en nuestro caso el "ftdi_sio.ko" al teléfono en algún directorio conocido por medio del módulo ADB, para esto se conectó el teléfono al computador y se ejecuta el comando:

$ adb push drivers/usb/serial/ftdi_sio.ko /sdcard/

Una vez conectado el teléfono al computador, se ingresa al terminal del teléfono con la ayuda del ADB y el comando "adb shell" y para la instalación del módulo se localizó el directorio en donde se guardó los módulos y se ejecuta:

$ insmod ftdi_sio.ko

Una vez realizado todo el proceso de instalación del archivo "ftdi_sio.ko" el teléfono se habilitó para usar el módulo FTDI, y al conectar la radio al teléfono, en el directorio "/dev" que contiene los archivos de dispositivos que permiten la comunicación con el hardware que tengamos en el teléfono, apareció el archivo "ttyUSB0" que indicó que la radio estaba conectada al teléfono y podía empezar a transmitir o recibir datos.

8. DISEÑO DE LA APLICACIÓN

El primer diseño consta de dos procesos: uno en "java" y uno en "C", que se comunican mediante el JNI, el proceso en "java" llama a una función nativa escrita en lenguaje "C" que abre un puerto mediante el módulo FTDI hasta obtener un dato, mientras el proceso "Java" espera la respuesta de esta función, una vez que hay respuesta el proceso en "Java" continua con su ejecución.

El problema que se presenta en esta versión es que al llamar a la función en lenguaje "C" el proceso en "Java" permanece bloqueado hasta recibir una respuesta, y al estar la interfaz de usuario en este proceso "Java" esta queda bloqueada. Este diseño no llegó a ser la solución debido al bloqueo de la interfaz de usuario. La primera versión del diseño se muestra en la figura 5.

El segundo diseño de la aplicación fue la versión final y consta al igual que el primero de 2 procesos uno en "Java" y otro en "C" que se comunican mediante un socket, el proceso en "Java" contiene la interfaz del mapa, el proceso en "Java" llama a un método nativo en lenguaje "C" que se encuentra implementado en una librería nativa y que contiene el archivo llamado "native.c", en este archivo se encuentra el código del proceso en "C".

El método nativo crea un hilo en donde se realiza todo el proceso de lectura de datos mediante el módulo ftdi, luego el método finaliza sin retornar valor alguno al proceso en "Java", es decir retorna a la interfaz donde se puede manipular el mapa sin bloquear la aplicación.

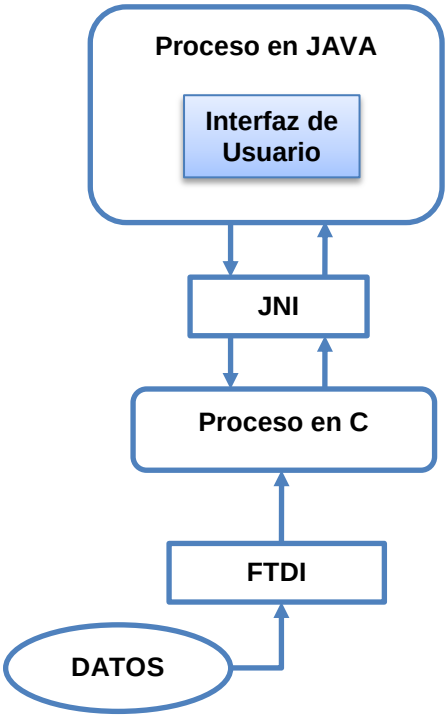

Figura 5.- Primer Diseño de la Aplicación

El hilo permanece en ejecución y accede al dispositivo a través del archivo "ttyUSB0" para poder realizar la lectura de los datos que llegaban al puerto. Para comunicar los procesos entre "C" y "Java" se utiliza un socket servidor en el proceso "C", en este socket se envían los datos leídos del archivo "ttyUSB0", y para recibir los datos en el proceso "Java" se utiliza un socket cliente pero que se conecta al mismo teléfono es decir al "anfitrión local" (localhost).

El hilo en el proceso "C" envía los datos de las coordenadas de manera constante, por este motivo el proceso "Java" debe tener una tarea que tome esos datos de la misma manera, para esto se utiliza un proceso asíncrono. El proceso asíncrono contiene al socket que recibe los datos, luego los depura y envía las coordenadas a la interfaz de usuario que procede a mostrarlas en el mapa.

El problema que se presentaba en la primera versión ya no se daba en la segunda versión ya que al crear un hilo la ejecución continuaba sin bloquear el proceso en "Java", y los datos se transmitían por medio del socket de comunicación, siendo esta versión la solución final del diseño. El segundo diseño de la aplicación se muestra en la figura 6.

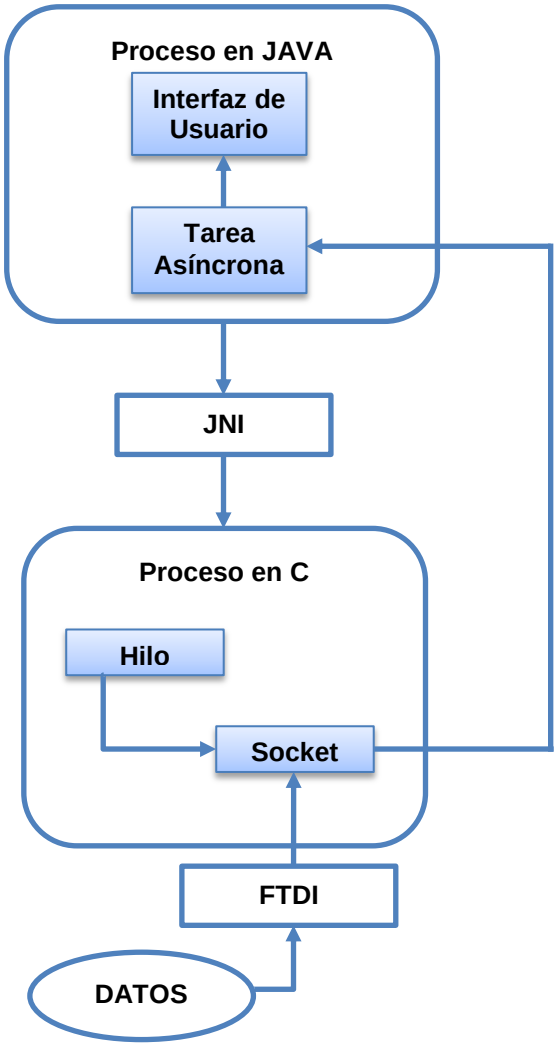

Figura 6.- Segundo Diseño de la Aplicación

9. FUNCIONAMIENTO DE LA APLICACIÓN

Al iniciar la aplicación además de presentar la pantalla principal, se obtienen los accesos de superusuario y se configura los permisos de lectura y escritura para el archivo "ttyUSB0". Después que se obtiene los permisos se accede a la pantalla del mapa al presionar el botón comenzar. Una vez dentro se puede ver el mapa de google, un botón para cambiar el tipo de mapa que se presenta y otro botón para iniciar y detener la toma de datos.

Al momento de presionar el botón iniciar se espera a que las tramas con las coordenadas geográficas empiecen a llegar. La trama está compuesta de los valores de coordenadas, la información del punto geográfico y el checksum al final de la trama, el

cheksum es un código hexadecimal que se calcula mediante una operación exor entre cada uno de los caracteres.

Se procede a calcular el checksum de la trama recibida y se lo compara con el checksum recibido. Si los valores son diferentes se considera una trama inválida y se descarta, quedando a la espera de la próxima trama.

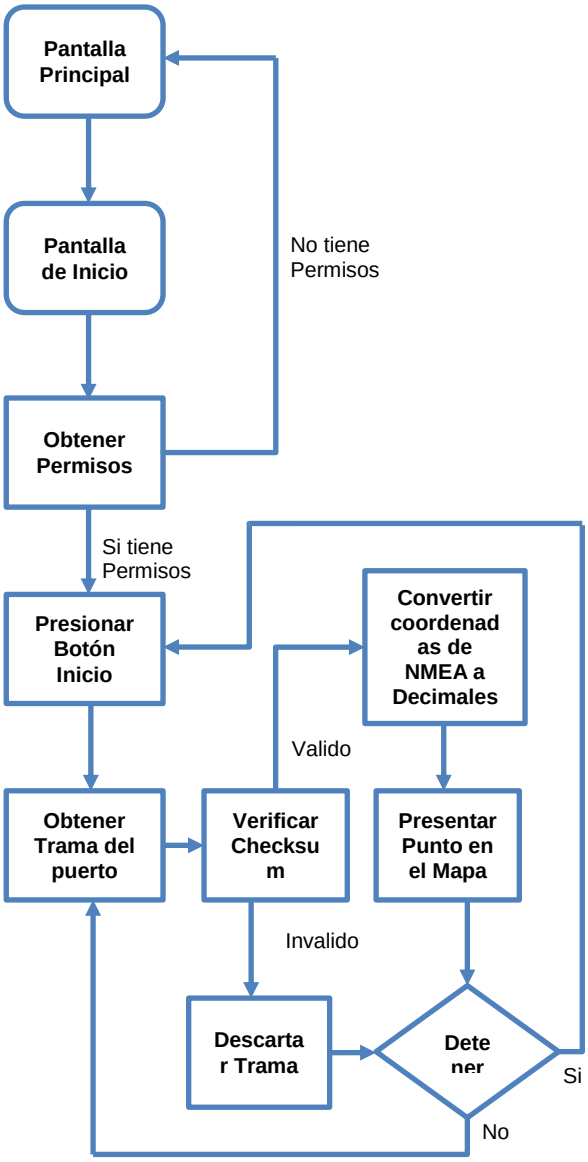

Figura 7.- Funcionamiento de la Aplicación

Si los checksum coinciden la trama es válida y se procede a transformar las coordenadas recibidas que están en formato NMEA a formato Decimal, luego se dibuja el punto en el mapa y se queda a la espera de la próxima trama.

En todo momento de la ejecución si se presiona el botón detener cualquier trama recibida no será dibujada y finalizara el proceso de obtención de datos. La figura 7 muestra el funcionamiento de la aplicación desde la captura de las tramas hasta la presentación del punto en el mapa.

10. EXPERIMENTO

Para realizar el experimento se conectó una de las radios Xbee a nuestra computadora portátil, esta funcionó como servidor y se encargó de transmitir las tramas con las coordenadas geográficas. La otra radio se conectó al teléfono y esta se encargó de recibir las coordenadas. Para medir la efectividad de las radios se modificó la frecuencia del envió de datos y la distancia en que se colocó las radios. Los datos se tomaron dentro de un domicilio, en donde las mediciones obtenidas a 1 y 3 metros fueron a línea vista y las mediciones obtenidas a 6, 15, 20 y 25 metros fueron entre paredes de concreto. Las diferentes frecuencias de transmisión a las que se enviaron los datos fueron de 1, 3, 5 y 10 segundos. En la figura 8 se pudo observar que la perdida de paquetes se incrementó conforme la distancia aumentó. Adicionalmente se pudo notar que a los 20 metros y a una frecuencia de transmisión de 1 segundo hubo una perdida mayor de paquetes, esto pudo haberse dado debido a la ubicación del receptor en ese momento, ya que realizamos las pruebas dentro de una casa.

Podemos concluir en base a los datos obtenidos, que la perdida de datos es proporcional a la distancia, y las variaciones de datos a distancia de 20 metros pudo haberse dado por el lugar donde se colocó el teléfono y por ende la radio, se debe tener en cuenta que las pruebas se realizaron dentro de una casa. También se observa que a mayor frecuencia existe menos perdida de datos y a distancia menores a 6 metros la perdida de datos en nula.

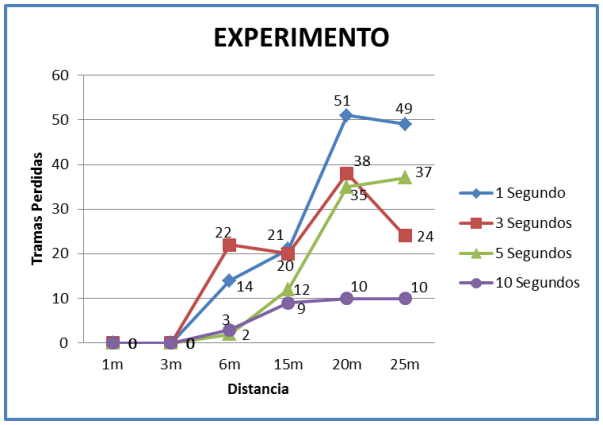

Figura 8.- Perdida de paquetes a diferentes frecuencias

11. CONCLUSIONES

1. Las pruebas fueron realizadas en un ambiente interior, y como era de esperar, la interferencia de equipos electrónicos (televisores, teléfono inalámbrico y radio) afectó la potencia de la señal emitida por la radio conectada al computador; esto se dedujo debido a que al momento de realizar una prueba de envió de datos en un laboratorio donde la única interferencia pudo provenir de la red inalámbrica (WIFI), a una distancia aproximada de 6 metros entre las radios, no se presentó perdida de información alguna.

2. Después de realizar las pruebas en un ambiente cerrado, y analizar la información obtenida en los experimentos, se determinó que se perdió el 34% de las tramas en promedio, de las 100 enviadas, si nos basamos en los picos de cada una de las curvas. En comparación al promedio de perdida de paquete proporcionados por las especificaciones de las radios utilizadas, que es de alrededor del 1%, el rendimiento de las radios no es el esperado, pero se debe recordar que el ambiente de las pruebas no era el óptimo, debido a la interferencia y obstáculos presentes durante el experimento.

3. La creación de un hilo a nivel del núcleo del sistema operativo Android, resolvió el problema del bloqueo de la interfaz de usuario que se tuvo en el primer diseño de la aplicación y por ende la obtención de tramas, ya que ahora la aplicación se encarga de procesar la trama obtenida y de seguir leyendo datos del puerto de manera simultánea. En caso de que el teléfono tuviera un procesador de múltiples núcleos, el procesamiento en paralelo sería posible, lo que mejoraría aún más la obtención de tramas a frecuencias más altas de envió de datos.

12. RECOMENDACIONES

1. Habilitar el módulo FTDI fue la parte más complicada en el desarrollo de este trabajo ya que se tuvo que buscar el núcleo apropiado para el tipo de teléfono que se tenía, y una vez encontrado había que configurarlo de tal que se pueda instalar en el teléfono sin que se produzca ningún error.

2. Para poder tener acceso a un dispositivo mediante el módulo FTDI no basta con instalar este módulo, también se debe revisar que la versión del núcleo de Android que se tiene tenga soporte OTG es decir se pueda utilizar el teléfono como un anfitrión.

3. El desarrollo de la aplicación inicialmente fue considerado para otras radios, las cuales no estuvieron bajo nuestra responsabilidad, lo que implicó que se realicen experimentos y cambios en el funcionamiento de las mismas. Lógicamente esto afectó la ejecución de nuestra aplicación, lo cual represento pérdida de tiempo en determinar el problema y solucionarlo. Se debe trabajar con recursos dedicados, o que no cambien su funcionamiento básico.

4. Las limitaciones del hardware de los teléfonos actuales deben ser revisadas antes de querer modificar el núcleo de un sistema operativo. A pesar de que se indicaba que la versión utilizada en nuestro teléfono de prueba inicial soportaba estos cambios, las limitantes del hardware impidieron su funcionamiento.

5. Debemos tener cuidado al modificar el núcleo de Android con Odín, ya que si se interrumpe el proceso de modificación ya sea porque el teléfono se descargue o se desconecte, dicho teléfono quedara inutilizable.

13. REFERENCIAS

[1] Alejandro Nieto Gonzalez, "¿Qué es Android?", http://www.xatakandroid.com/sistema-operativo/que-es-android, Fecha de Publicación: 8 de Febrero del 2011

[2] Blanco Lezcano y Jent Chong, Facultad de Electrotecnia y Computación Universidad Nacional de Ingeniería, "Android Operating System", http://electrouni.files.wordpress.com/2010/12/android-os.pdf, 16 de Noviembre del 2010

[3] comScore, "comScore Reports June 2013 U.S. Smartphone Subscriber Market Share", http://www.comscore.com/Insights/Press_Releases/2013/8/comScore_Reports_June_2013_U.S._Smartphone_Subscriber_Market_Share, Fecha de Publicación: 7 de Agosto del 2013

[4] Mobileburn, "What is NFC?", http://www.mobileburn.com/definition.jsp?term=NFC, 12 Junio del 2013

[5] Paul Paliath, "Android 4.1 JellyBean vs iOS 6 vs Windows Phone 8 – TheUltimateComparison", http://www.redmondpie.com/android-4.1-jelly-bean-vs-ios-6-vs-windows-phone-8-the-ultimate-comparison/, Fecha de Consulta: 6 de Julio del 2012

[6] AndroidDevelopers, "Glosario Android", http://developer.android.com/guide/appendix/glossary.html, Fecha de consulta: Enero 2013

[7] Condesa, "La máquina virtual Dalvik", http://androideity.com/2011/07/07/la-maquina-virtual-dalvik/, Fecha de publicación: Julio 7 del 2011

[8] Android: a programmer's guide; Jerome

DiMarzio; McGraw-Hill; 2008;

[9] Xianzhong Zhu, "Introduction to Android JNI development Using NDK", http://dotnetslackers.com/articles/net/Introduction-to-Android-JNI-development-Using-NDK-Part-1.aspx, Fecha de Publicación: 16 de diciembre del 2011

[10] Android Architecture, http://elinux.org/Android_Architecture, Fecha de Consulta: Junio del 2013

[11] androideity, "Arquitectura de android", http://androideity.com/2011/07/04/arquitectura-de-android/, Fecha de publicación: Julio 4 del 2011

[12] Hello, Android: Introducing Google's Mobile Development Platform; Ed Burnette; Pragmatic Bookshelf; 2010;

[13] FutureTechnologyDevices International Limited, "Opciones para perifericos en Android", http://www.ftdichip.com/Support/Documents/White_Papers/WP_003_Android_Peripheral_Options.pdf, 11 Febrero del 2013