UNIVERSIDAD AUTONOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO TEXCOCO

INTEGRACIÓN DE PLATAFORMAS DE HARDWARE Y SOFTWARE LIBRE EN DISPOSITIVOS MÓVILES PARA EL DESARROLLO DE PROCESOS DE AUTOMATIZACIÓN Y ANÁLISIS DE INFORMACIÓN EN TIEMPO REAL

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

SERGIO CONTRERAS MURILO

TUTOR ACADÉMICO
DR. OZIEL LUGO ESPINOSA

TUTORES ADJUNTOS
DR. JOEL AYALA DE LA VEGA
DR. ALFONSO ZARCO HIDALGO

TEXCOCO, ESTADO DE MÉXICO NOVIEMBRE 2014

UAEM | Universidad Autónoma del Estado de México

CUTex

DICTÁMEN DE AUTORIZACIÓN Y OBTENCIÓN DE GRADO DE MAESTRÍA

Texcoco, Méx. , a 18 de Noviembre del 2014

Título del proyecto:

Integración de plataformas de hardware y software libre en dispositivos móviles para el desarrollo de procesos de automatización y análisis de información en tiempo real.

Tesista:

Sergio Contreras Murillo

Dictamen: **No. de revisión: 4**

- ☐ **Rechazado**
- ☐ **Sujeto a modificaciones**
- ☐ **Aceptado, condicionado**
- ☒ **Aceptado**

POSGRADO TITULACION RECIBIDO
Por Adriana
Texcoco, Méx., a 18 de XI del 14

Observaciones generales:

Aceptado para la impresión
Aceptado para la defensa de grado

Tutor Adjunto	Tutor Académico	Tutor Adjunto
Dr. Joel Ayala de la Vega	**Dr. Oziel Lugo Espinosa**	**Dr. Alfonso Zarco Hidalgo**

Agradecimientos

Este trabajo está dedicado a mis padres y a mis hermanos que siempre han sido parte de todas las metas que he alcanzado y que con su cariño y ejemplo me guían en las tareas que emprendo.

Agradezco también a mi tutor académico, el Dr. Oziel Lugo Espinoza que con sus consejos y apoyo dio forma a este trabajo de investigación. A los doctores Joel Ayala de la Vega y Alfonso Zarco Hidalgo por su colaboración y su confianza en este proyecto. A todos los profesores de la maestría e ingeniería y al personal del C.U. que con su labor diaria contribuyen a la formación profesional y humanista de los alumnos.

Al CONACYT y COMECYT por el patrocinio sin el cual no hubiera podido terminar mis estudios.

A mis compañeros de la maestría que compartieron conmigo su experiencia, conocimiento, opiniones, tiempo, esfuerzo y hasta su comida durante estos 2 años y medio.

También dedico mi tesis a mis sobrinos Martin, Kenia y Toñito; y a Jonatán y Carlos que quieren ser escritores, y ya ven que aunque sea una tesis pero yo también ando escribiendo.

En 2015 el C.U. cumple 20 años, recuerdo que cuando iba a la secundaria solía esperar leyendo sentado en un árbol afuera de la entonces UAP Texcoco a que mi hermana Rosa saliera de sus clases, la esperaba leyendo "*Clemencia"* y los *"cuentos de invierno"* de Ignacio Manuel Altamirano. En ese entonces en que la universidad no tenía más que un edificio después de tres años de no tener ninguno, me quedaba muy claro que una universidad no la definen sus instalaciones o su ubicación, la crean los alumnos, los maestros y todos los que trabajan en ella.

En este trabajo se presenta una propuesta de plataforma que integra hardware y software para dispositivos móviles que permite el desarrollo ágil de prototipos de interés científico. Se aborda, desde la perspectiva de ingeniería de software, el proceso para conectar sensores a un celular o tableta electrónica y la ejecución de software estadístico.

La arquitectura de aplicación móvil propuesta aprovecha las cualidades de movilidad y autonomía de los dispositivos móviles para ayudar al trabajo en campo de investigadores y personal especializado que realiza tareas de automatización, medición y monitoreo.

Se dio prioridad al uso de plataformas de software y hardware libre de bajo costo que permiten adaptar fácilmente la plataforma a otros escenarios de desarrollo, es decir, se permite adaptar la plataforma de acuerdo a las necesidades del problema a resolver.

En la primera parte del presente documento se describe el proceso que permite la captura automática de información en campo mediante la conexión directa de instrumentos de medición o sensores al dispositivo. Los datos recabados son analizados en tiempo real gracias a la instalación nativa de un lenguaje de procesamiento estadístico en el dispositivo electrónico.

La investigación es realizada desde un punto de vista arquitectónico y se aborda por tanto la aplicación de buenas prácticas y patrones de diseño modernos del desarrollo de software y las consideraciones técnicas derivadas de la interacción con sensores electrónicos analógicos y digitales.

La instalación nativa de R en el sistema operativo Android permite que desarrolladores creen versiones móviles de sus aplicaciones para escritorio trasladando de forma transparente la lógica de negocios.

En la segunda parte del documento se presentan una aplicación para equipos de escritorio que, al migrarse a una versión para dispositivos móviles, hace uso de la plataforma.

La aplicación móvil sirve para obtener el área, largo y ancho de hojas alargadas con tan solo tomarles una foto contra un fondo claro. La aplicación utiliza el lenguaje R para tratar digitalmente las imágenes tomadas a una distancia conocida gracias a un sensor ultrasónico

conectado al dispositivo móvil. La aplicación puede ser utilizada por agricultores para el monitoreo del crecimiento de cultivos.

La versión stand-alone de la misma aplicación analiza imágenes obtenidas desde un escáner con la misma lógica de negocios codificada en R. También le permitirá al usuario navegar las imágenes tomadas de la aplicación móvil.

Contenido

Índice de figuras

Ejemplos presentes en la tesis

Antecedentes

Debido a la heterogeneidad de hardware y poca capacidad de almacenamiento de los primeros dispositivo móviles, las aplicaciones solían (y aun hoy lo hacen) conectarse a servidores remotos para el análisis y persistencia de su información.

Las redes sociales, los contenidos multimedia sobre demanda, revistas, periódicos, diccionarios, traductores, reportes meteorológicos y de noticias, etc.; son ejemplos de clientes móviles que se conectan a un servicio web.

También en el campo científico los dispositivos móviles se descartaban para analizar y almacenar la información de manera nativa.

Por ejemplo, El cliente móvil “Matlab Mobile©” se conecta a través de internet a un servicio en la nube. Para el uso de la aplicación es necesario haber adquirido la licencia de uso para escritorio (**Ilustración 1**).

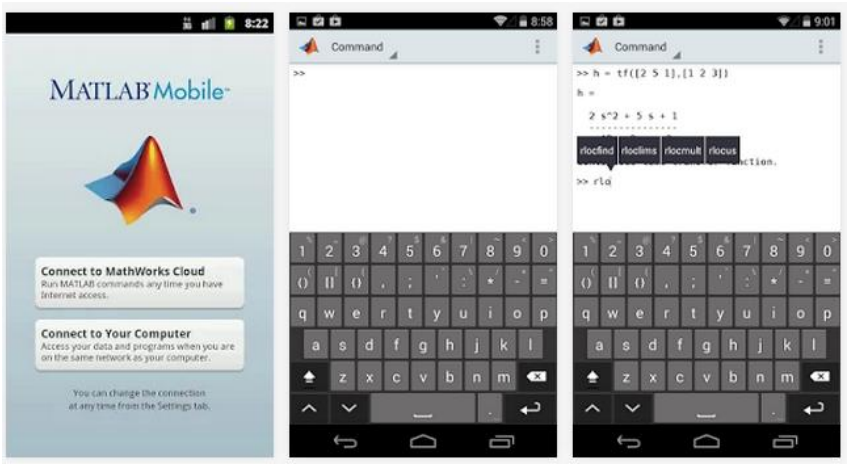

Ilustración 1 Matlab mobile

Al momento de desarrollar una aplicación móvil nos encontraremos con que cada sistema operativo ejecuta sus propias aplicaciones desarrolladas en lenguajes diferentes. Incluso entre versiones del mismo SO hay problemas de compatibilidad.

Algunas empresas e instituciones optan por no realizar aplicaciones móviles sino optimizar sus sitios corporativos para garantizar su funcionalidad en el mayor número de plataformas posibles.

Primefaces© (Primefaces) es un framework de desarrollo de aplicaciones Java Server Faces cuya suite incluye componentes optimizados para su uso en dispositivos móviles. La **Ilustración 2** muestra un sitio web optimizado.

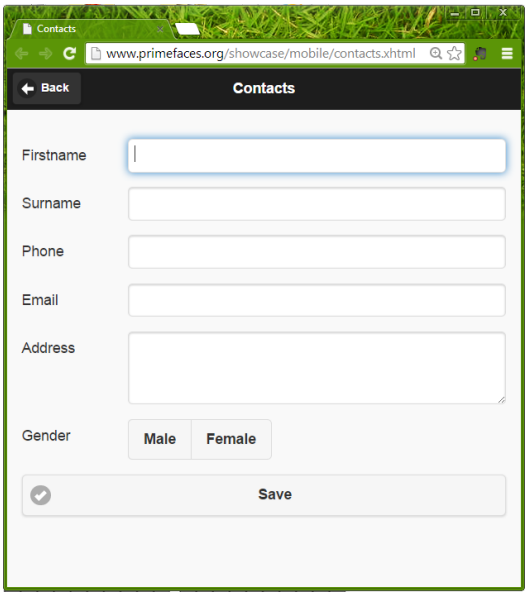

Ilustración 2 Primefaces mobile

Sin embargo, un esquema que depende de internet para el procesamiento y almacenamiento de información se dificulta en exteriores donde las condiciones meteorológicas y geográficas contravienen las características de movilidad y autonomía con que celulares y tabletas son diseñados.

Por otro lado, la capacidad de procesamiento y almacenamiento de celulares y tabletas que ejecutan el sistema operativo Android se ha incrementado al igual que su popularidad. Los equipos de gama alta que se encuentran actualmente disponibles cuentan con procesadores de varios núcleos, gran espacio de almacenamiento y la capacidad de operar con componentes de hardware compatibles.

Estas características pueden ser aprovechadas por desarrolladores y usuarios para procesar información in situ de forma confiable y eficaz.

La integración de plataformas de software y hardware libre para dispositivos móviles que procesen en tiempo real información recabada mediante sensores conectados directamente facilitará a quien realice trabajo en campo el análisis de información y la toma de decisiones.

Son cada vez más los ejemplos del uso de los dispositivos móviles en la educación, medicina, agricultura, industria e investigación. Por ejemplo, La aplicación SeedStar© (John Deere, 2014) de la empresa de maquinaria agrícola John Deere© recaba y monitorea información en tiempo real sobre la eficacia con la que las semillas son plantadas (**Ilustración 3**).

Ilustración 3 SeedStar Mobile

El uso en campo de los dispositivos móviles ha generado que las compañías presenten opciones resistentes al agua o polvo. La Tablet de uso rudo E1 fieldbook (**Ilustración 4**) es además resistente a caídas de hasta 1.8 metros de altura.

Ilustración 4 E1 Fieldbook

El uso masivo de teléfonos inteligentes que ejecutan sistemas operativos inició apenas la década pasada. Es aún más reciente su uso con propósitos científicos. Específicamente, el uso de las tabletas electrónicas ha aumentado desde que en el 2010 se lanzara la primera de ellas, el Ipad© de la compañía Apple© (Apple inc.). De acuerdo al *"Segundo estudio de usos y hábitos de dispositivos móviles"* (Milward Brown, 2013) 84% de los mexicanos cuenta con algún dispositivo móvil, y de ellos el 40% poseen un Smartphone.

El estudio refiere que el 87% de usuarios portan diariamente su dispositivo y que nunca lo apagan. Por otro lado, sólo el 35% de los dueños de un smartphone cuenta con acceso a internet mediante una red de datos 3G.

Android© (Android Open Source Project), IOS© (Apple inc.) y Windows Phone© (Microsoft) se dividen el mercado de los Sistemas operativos para dispositivos inteligentes. En el segundo trimestre del 2014 Android lidera las ventas de nuevos dispositivos con el 84.7% seguido de IOS con 11.7% y Windows Phone con tan sólo el 2.5%. (IDC, 2013). (**Ilustración 5**)

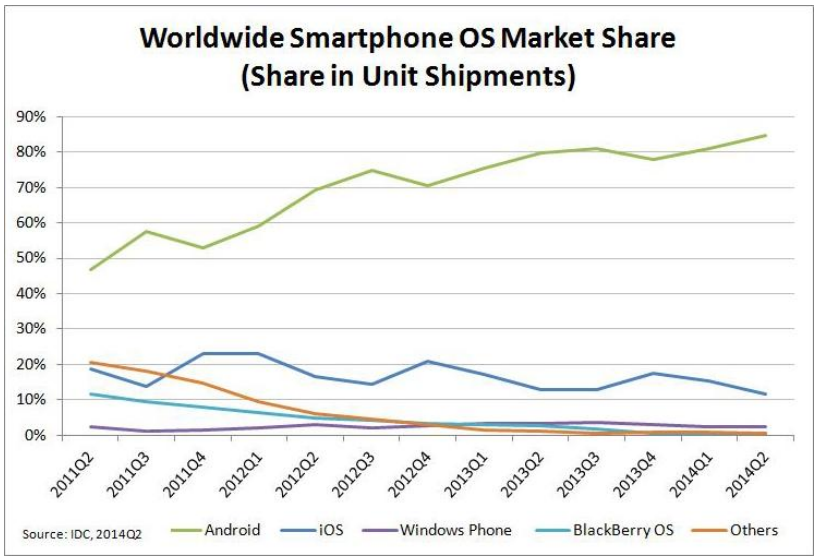

Ilustración 5 Cuota de mercado mundial de SO's para teléfonos inteligentes (IDC, 2013)

En la revisión bibliográfica no se encontraron antecedentes del uso de lenguajes de programación estadística en proyectos de interés científico en dispositivos móviles pero el lenguaje R (The R project) tiene versiones específicas para Android e IOS.

R, SAS y Matlab son los lenguajes estadísticos más utilizados, en ese orden, de acuerdo al índice TIOBE (**Ilustración 6**). *El índice de la comunidad de programadores "TIOBE" es un indicador de la popularidad de los lenguajes de programación. El índice se actualiza una vez al mes. Las calificaciones se basan en el número de ingenieros cualificados en todo el mundo, cursos y proveedores terceros. Los motores de búsquedas populares como Google, Bing, Yahoo, Wikipedia, Amazon, Youtube y Baidu son usados para calcular las clasificaciones.* (TIOBE, 2014)(**Ilustración 7**)

Posición Oct 2014	lenguaje	popularidad	cambio
15	R	1.523%	+0.97%
22	SAS	0.679%	
27	MATLAB	0.608%	

Ilustración 6 Posiciones de R, SAS y MATLAB en el Índice TIOBE 2014

"R es un lenguaje y entorno para cómputo estadístico y graficación, similar al lenguaje S desarrollado originalmente en los laboratorios Bell. Es una solución de código abierto para el

análisis de información que es mantenido por una activa y numerosa comunidad alrededor del mundo" (Kabacoff, 2011). El lenguaje R es multiplataforma y posee distribuciones específicas para sistemas operativos de dispositivos móviles tales como Android e IOS© (Apple inc.).Otras alternativas comerciales, como SAS© y Matlab© (Matlab, 2013), no cuentan con distribuciones adaptadas a sistemas operativos de dispositivos móviles.

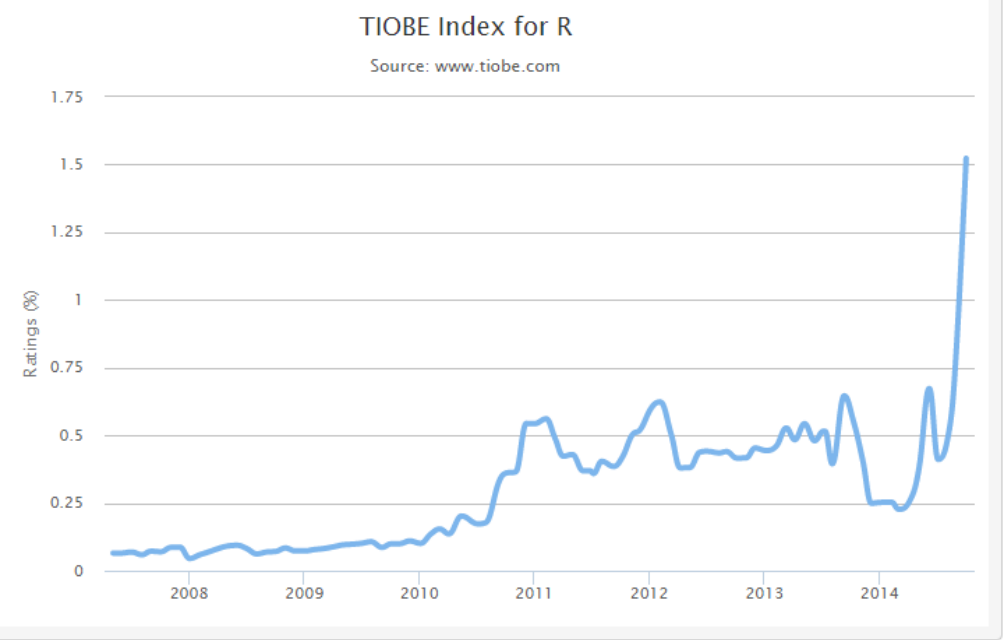

Ilustración 7 TIOBE index for R

R cuenta con la ventaja de ser gratuito mientras que SAS y MATLAB no. Además R es interoperable con aplicaciones para el SO Android por que se realizan con Java (SAS y MATLAB no son interoperables con aplicaciones para el SO Android).

R es ampliamente utilizado en el ámbito académico y de investigación. Por ejemplo, el proyecto Ibfieldbook (Lugo-espinosa, 2013) utiliza el lenguaje R para el análisis de la herencia de rasgos fenotípicos en granos de maíz, trigo y arroz tales como el tamaño, color, producción o velocidad de crecimiento.

En 2011 Google anuncio el lanzamiento del Android Open Accesory Kit (Android Open Source Project, 2012) que permitiría conectar piezas de hardware a los dispositivos móviles. El ADK

provee una implementación de un Accesorio USB basada en la plataforma de desarrollo de prototipos electrónicos Arduino (Arduino).

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios (wikipedia).

Arduino es igualmente utilizado desde proyectos de domótica que en instalaciones artísticas.

La conexión de un sensor a un dispositivo móvil depende de que el fabricante haga públicos sus protocolos de comunicación o ponga a disposición del público en general la biblioteca necesaria.

En este trabajo se menciona cómo utilizar la interface Arduino para recabar información desde sensores y cómo utilizar el lenguaje R en el dispositivo móvil para tratar la información.

Este proyecto de investigación busca desarrollar una propuesta de plataforma de hardware y software libre que explore el rendimiento de un equipo móvil en el procesamiento estadístico en tiempo real de información recabada por sensores. Se implementarán aspectos arquitectónicos y patrones de diseño modernos con el fin de que sea fácilmente adaptable a otros escenarios.

La plataforma propuesta permite la creación de versiones móviles de aplicaciones de escritorio que usan el lenguaje R sin que sea necesario programar de nuevo la lógica de negocios. Esto reduce el costo y tiempo de desarrollo de prototipos móviles de interés científico. Además, contempla la interoperabilidad con sensores conectados directamente al celular o tableta con el objetivo de facilitar la captura de información para los profesionales que llevan a cabo tareas de medición y monitoreo en campo.

Planteamiento del problema

Las mediciones manuales pueden propagar errores en una investigación. El personal que realiza tareas de medición o monitoreo en campo requiere herramientas portátiles, robustas y eficaces que sean capaces de tomar lecturas y analizar en tiempo real la información recabada.

Al mismo tiempo, hay que observar la correspondencia con los sistemas de escritorio que integrarán la información recabada.

Estructuras que dependen del acceso a internet por parte del usuario no pueden realizar el análisis en tiempo real de la información y su uso se dificulta en lugares remotos o con condiciones adversas de clima.

De no ser posible utilizar un lenguaje estadístico en dispositivos móviles, sería necesario reescribir toda la lógica de negocios de estadística o inteligencia artificial de desarrollos de escritorio de los que se desee desarrollar una versión para celulares. Esto incide directamente en el costo y tiempo de desarrollo de la versión para dispositivos móviles de una aplicación.

Objetivo general

Diseñar una plataforma de software y hardware libre para dispositivos móviles que permita analizar información recabada en campo mediante sensores conectados directamente al dispositivo.

Objetivos específicos

- Desarrollar el módulo de recepción de señales desde sensores utilizando la plataforma gratuita de código abierto “Arduino”.
- Desarrollar el módulo de tratamiento de la información mediante la implementación en el dispositivo móvil del paquete estadístico “R”.
- Desarrollar una estrategia de persistencia de la información.
- Desarrollar la interfaz gráfica de usuario para el sistema operativo Android que oculte al usuario la complejidad del problema y le facilite el análisis de resultados y la toma de decisiones.
- Validar los resultados obtenidos con la integración de las plataformas mediante la solución de un problema en particular.

Supuesto

La integración de una plataforma de software y hardware libre para dispositivos móviles que procese en tiempo real información recabada mediante sensores, a través de un lenguaje de procesamiento estadístico, facilitará la toma de decisiones a investigadores que realizan trabajo en campo.

Justificación

1. La utilización de software gratuito reduce los costos de desarrollo de prototipos de interés científico, a la vez que al ser de código abierto se favorece su integración en nuevas plataformas.
2. Las aplicaciones nativas funcionan de acuerdo con el objetivo de portabilidad y autonomía de los dispositivos móviles.
3. Las aplicaciones nativas que no se comunican con servidores para el análisis o persistencia de información procesan en tiempo real los datos recabados sin pausas generadas de la comunicación vía internet.
4. El desarrollo de una estrategia de conexión de sensores a dispositivos móviles permitirá que el levantamiento de lecturas en campo no requiera una captura manual, lo que retribuirá en velocidad y exactitud.
5. La plataforma presentada utiliza hardware y software gratuito o de bajo costo presente en el entorno académico y científico, lo que facilita su adaptación en problemas heterogéneos
6. La reducción de costos y la mejora en sus componentes ha aumentado el uso de plataformas móviles. También se han incrementado la capacidad de memoria y procesamiento de celulares y tabletas.

Marco teórico

Introducción

En esta sección se mostrarán los elementos teóricos de los componentes utilizados para el desarrollo del proyecto. Los elementos a describir son:

- *Los sistemas de Microcomputadora embebidos.*
- *La arquitectura de una tableta y su interface con una microcomputadora embebida.*
- *El Sistema Operativo para dispositivos móviles Android (Basado en Unix).*
- *El lenguaje de programación estadístico R.*
- *La orquestación de un sistema móvil de interés científico.*

Los sistemas de microcomputadora embebidos

Un sistema de microcomputadora embebido incluye una microcomputadora conectada a dispositivos mecánicos, químicos y eléctricos, programada para un propósito dedicado y empacada como un sistema completo. Cualquier sistema eléctrico, mecánico o químico que incluya entradas, decisiones, cálculos, análisis y salidas es candidato para implementarse como un sistema embebido. Los sensores eléctricos, mecánicos y químicos reúnen información. Las interfaces electrónicas convierten las señales de un sensor a una forma que acepta la microcomputadora. El software de la microcomputadora efectúa las decisiones, los cálculos y los análisis necesarios. Las señales electrónicas de la interfaz convierten la salida de la microcomputadora a la forma deseada. Se utilizan actuadores para crear las salidas mecánicas o químicas.

Normalmente el software de los sistemas embebidos solo resuelve una gama limitada de problemas. La microcomputadora se incorpora u oculta dentro del dispositivo. La memoria de solo lectura ROM es un tipo de memoria donde la información se programa o se graba dentro del dispositivo y los datos se conservan guardados incluso si se interrumpe la energía eléctrica.

En un sistema embebido el software se programa en la ROM y, por tanto, esta fijo. A pesar de ello, el software de mantenimiento (Verificar que el software funcione de manera correcta, actualizar el software, eliminar defectos, agregar funciones, extender las funciones a nuevas aplicaciones y modificar las configuraciones de los usuarios finales) es muy importante.

En contraste, un sistema de cómputo de propósito general tiene un teclado, un disco y una pantalla que se programan para una gran variedad de propósitos.

El término "sistema de microcomputadora embebido" se refiere a un dispositivo que tiene una o más microcomputadoras en su interior. En este contexto, la palabra "embebido" significa oculto en el interior para que nadie lo pueda ver.

*Una computadora es un dispositivo electrónico con un procesador, memoria y puertos de E/S. El procesador ejecuta el software, el cual efectúa operaciones predefinidas específicas. El procesador incluye registros, los cuales son memoria de alta velocidad. Una unidad lógica aritmética (ALU) para ejecutar funciones matemáticas; una unidad de interfaz de bus que se comunica con la memoria y los puertos de E/S, y una unidad de control para tomar decisiones (**Ilustración 8**).* (Valvano, 2004)

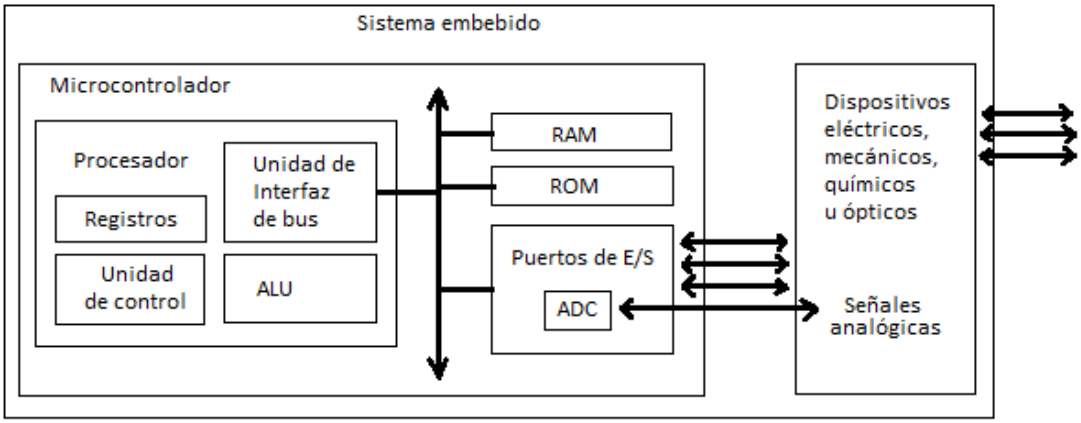

Ilustración 8 un sistema embebido incluye un microcontrolador que hace interfaz con dispositivos externos

Los sistemas en tiempo real riguroso deben ser completamente predecibles respecto al momento en que tienen lugar los eventos, esto es, procesamiento basado en marcos de tiempo. (booch, 1996).

La arquitectura de una tableta y su interface con una microcomputadora embebida

La **ilustración 9** muestra el diagrama de bloques completo de un dispositivo móvil. Por un lado, las aplicaciones móviles están acotadas a las limitaciones de almacenamiento y batería de los dispositivos que las ejecutan, mientras que, por otro lado, aprovechan las características especiales, como la presencia de sensores y cámaras. La conexión de sensores adicionales a un smartphone aumenta sus funcionalidades.

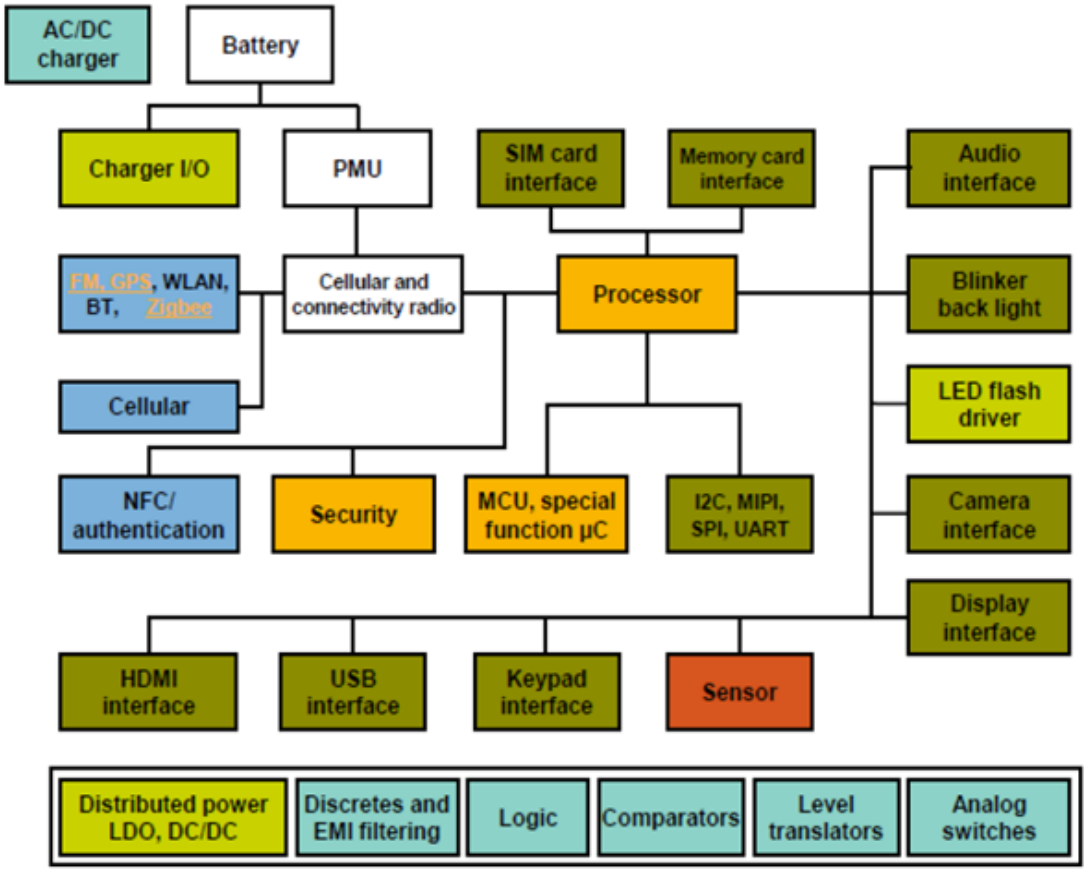

Ilustración 9 Diagrama de bloques de una tablet

Para conectar sensores a un dispositivo móvil puede utilizarse alguna de las plataformas de propósito general existentes en el mercado. En el caso del sistema operativo Android, se define un protocolo de comunicación con el dispositivo denominado *Android Open Accessory Protocol* (Android Open Source Project, 2011) y un kit de desarrollo, las Accesory Development Tools (Android Open Source Project).

"El kit de desarrollo de accesorios (ADK) es el punto de partida para los fabricantes de hardware en la construcción de accesorios para Android. Cada versión ADK se proporciona con las especificaciones del código fuente y de hardware para hacer que el proceso de desarrollo de sus propios accesorios sea más fácil.

Los accesorios de Android pueden ser estaciones de conexión para audio, máquinas de ejercicio, dispositivos de pruebas médicas personales, estaciones meteorológicas, o cualquier otro dispositivo de hardware externo que se suma a la funcionalidad de Android.

Los accesorios utilizan el protocolo ***AOA*** (Android Open Source Project, 2011) *para comunicarse con los dispositivos Android, a través de un cable USB o de una conexión Bluetooth"* (Android Open Source Project, 2012) *(Ilustración 9)*

Android soporta una gran variedad de periféricos USB y accesorios USB de dos maneras: USB accesory y USB host. En el modo ***Accesory***, el hardware USB externo actúa como Anfitrión. Esto le da la habilidad de interactuar con hardware USB a dispositivos Android que no tienen la capacidad de hacerlo. En el modo ***Host***, el dispositivo Android actúa como anfitrión y fuente de energía para el bus (**Ilustración 10**).

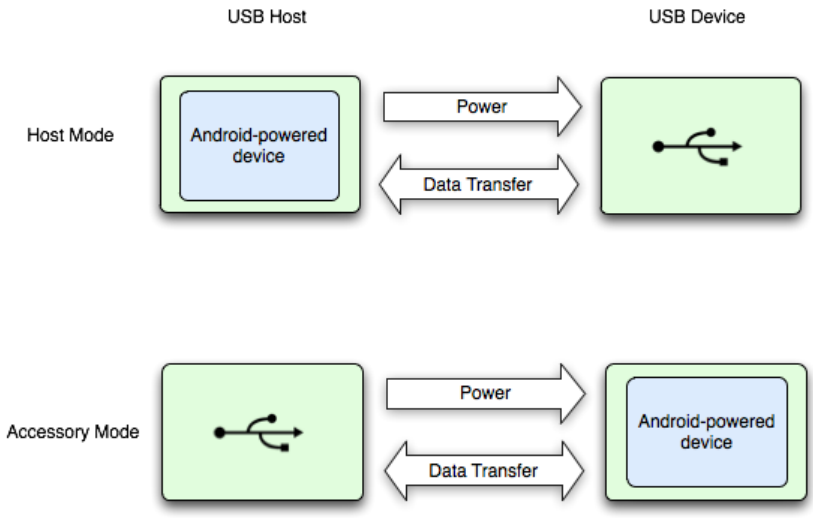

Ilustración 10 Modos Host y accesory de la conexion USB.

Tipos de conexión USB:

- **Accesory**: Estaciones de carga. Controles robóticos. Lectores de tarjetas. Equipo musical.
- **Host**: Cámaras digitales. Teclados. Ratones. Controles para videojuegos.

El protocolo AOA de Android se desarrolló para que fuera compatible con la plataforma Arduino. Las tabletas Arduino utilizan un microcontrolador programable de 8bits Atmel (Atmel, 2012). (**Ilustración 11**).

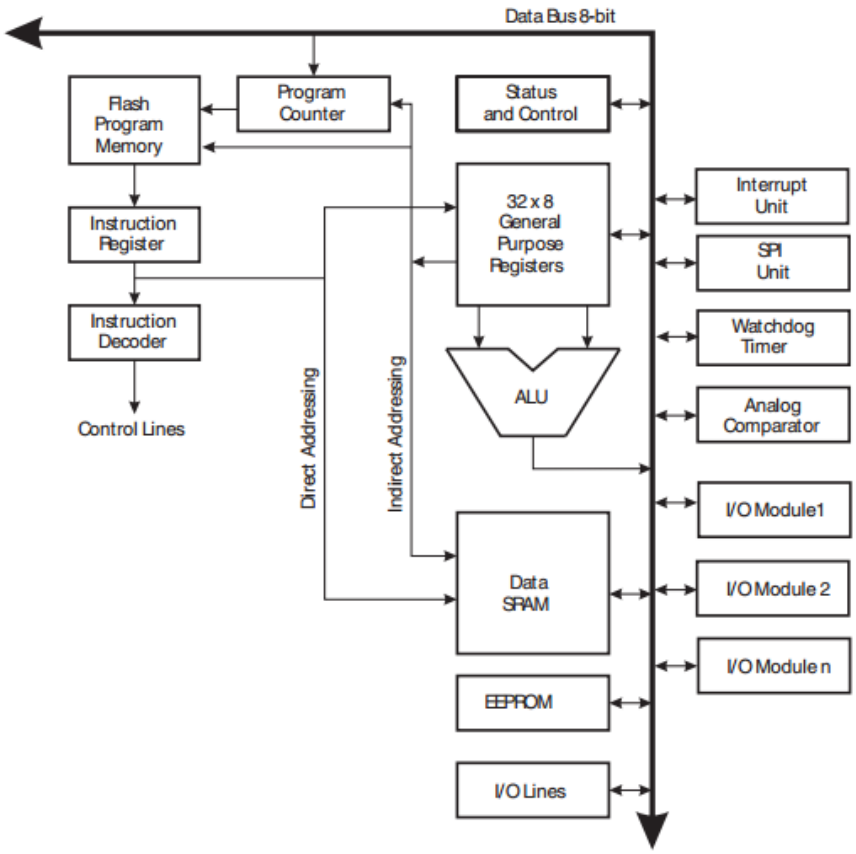

Ilustración 11 Diagrama de bloques del microcontrolador Atmel

El AVR utiliza el concepto de arquitectura Harvard - con buses separados para programas y datos. Mientras una instrucción es ejecutada, la siguiente es precargada desde la memoria del programa. Este concepto permite la ejecución de instrucciones en cada ciclo de reloj. La

memoria de programas es una memoria flash programable. Con algunas excepciones, las instrucciones AVR tienen un formato de 16 bits por palabra, lo que significa que cada dirección de memoria contiene una única instrucción de 16 bits.

Durante las interrupciones y las llamadas a subrutinas, la dirección de retorno del contador del programa (PC) es almacenada en la pila. La pila es efectivamente alojada en la memoria general SRAM, y consecuentemente el tamaño de la pila está limitado al tamaño total de la SRAM y al uso de la SRAM. Todos los programas deben inicializar el apuntador de la pila (SP) en la rutina de reinicio (antes de la ejecución de subrutinas o interrupciones).

Los 4000 bytes de información SRAM pueden ser fácilmente accedidos a través de los 5 diferentes modos de direccionamiento soportados en la arquitectura AVR.

Un módulo flexible de interrupciones tiene sus registros de control en el espacio I/O con una bandera de interrupción global adicional en el registro de estado (SR). Todos los distintos interruptores tienen un vector en la tabla de vectores de interrupción al inicio de la memoria del programa. Los diferentes interruptores tienen una prioridad de acuerdo a la posición de su vector de interrupción. Mientras más baja su posición, más alta su prioridad. (Atmel corporation, 1999)

La versión ADK de la tableta Arduino se conecta a los dispositivos móviles por medio de un controlador MAX3421E. El controlador MAX3420E USB-periférico/host contiene la circuitería digital y analógica necesaria para ser la interface entre el bus SPI (Interface serial de Periféricos) vía USB. (Maxim integrated, 2014) (**Ilustración 12**)

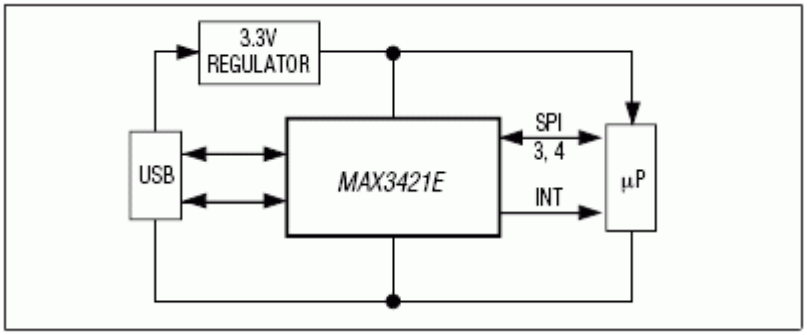

Ilustración 12. MAX3421E: circuito de aplicación típica

El Bus SPI (del inglés Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj. (Wikipedia, 2013)

Si bien pueden conectarse muchos sensores digitales y analógicos a una sola tableta Arduino, la comunicación recae en un único buffer de datos. Tanto Android puede hacer solicitudes a Arduino como Arduino desencadenar eventos en el SO. Android.

Para las señales analógicas, es significativo el valor preciso de la cantidad (voltaje, Angulo, de rotación, etc.) que porta la información. Las señales digitales son básicamente de naturaleza binaria (conducción / no conducción) y las variaciones en el valor numérico están asociadas con cambios en el estado lógico (verdadero/falso) de alguna combinación de interruptores. En un sistema electrónico digital típico, cualquier voltaje en el intervalo de +2 a +5 volts produce el estado de conducción, en tanto que las señales de 0 a +0.8 V corresponden a no conducción. De tal manera que si el voltaje es de 3 o 4 V, no tiene consecuencias. Se obtiene el mismo resultado y por lo tanto el sistema es bastante tolerante a los voltajes espurios de ruido que podrían contaminar la señal de información. (Doebelin, 2005)

Los sensores suelen tener un espacio de memoria en el que guardan una estructura de datos definida por el fabricante. Sólo si el fabricante hace públicas dichas especificaciones puede desarrollarse con C (Arduino se programa en C) una biblioteca para el uso del sensor.

El sistema operativo para dispositivos móviles Android

Android (Android Open Source Project) es un sistema operativo para celulares desarrollado por la *"**Open Handset Alliance**"*, una alianza comercial de 84 empresas de software y hardware telefónico lideradas por la compañía ***Google***© (Google). Está basado en ***Linux***© (Linux org.), es gratuito y de código abierto. Fue presentado en 2008 y para 2013 ya estaba presente en 900 millones de dispositivos.

Las aplicaciones para el sistema operativo Android se codifican en el lenguaje orientado a objetos Java. La máquina virtual "***Dalvik***" que ejecuta las aplicaciones móviles es diferente a la máquina virtual Java JVM definida para equipos de escritorio.

Dan Bornstein de la compañía Google© escribió Dalvik para optimizar el almacenamiento, ejecución y vida de la batería. Dalvik combina las clases Java generadas en uno o más archivos ejecutables con terminación .dex. (Komatineni, 2012).

En el núcleo de la plataforma Android está el kernel Linux responsable de los drivers, el acceso a recursos, administración de la energía, etc. El SO controla el Display, Camaras, teclado, Wi-Fi, la memoria flash, audio y la comunicación inter-procesos (IPC).

Por encima de él hay un gran número de bibliotecas C/C++ tales como OpenGL, WebKit, FreeType, Secure Sockets Layers (SSL), SQLite y la biblioteca de ejecución de C (libc).

La mayoría de las aplicaciones acceden a las bibliotecas a través de la VM Dalvik. La VM Dalvik está optimizada para ejecutar múltiples instancias de máquinas virtuales.

Las API's java de Android incluyen telefonía, ubicaciones, recursos, UI, proveedores de contenidos y administradores de paquetes. (Komatineni, 2012) (**Ilustración 13**).

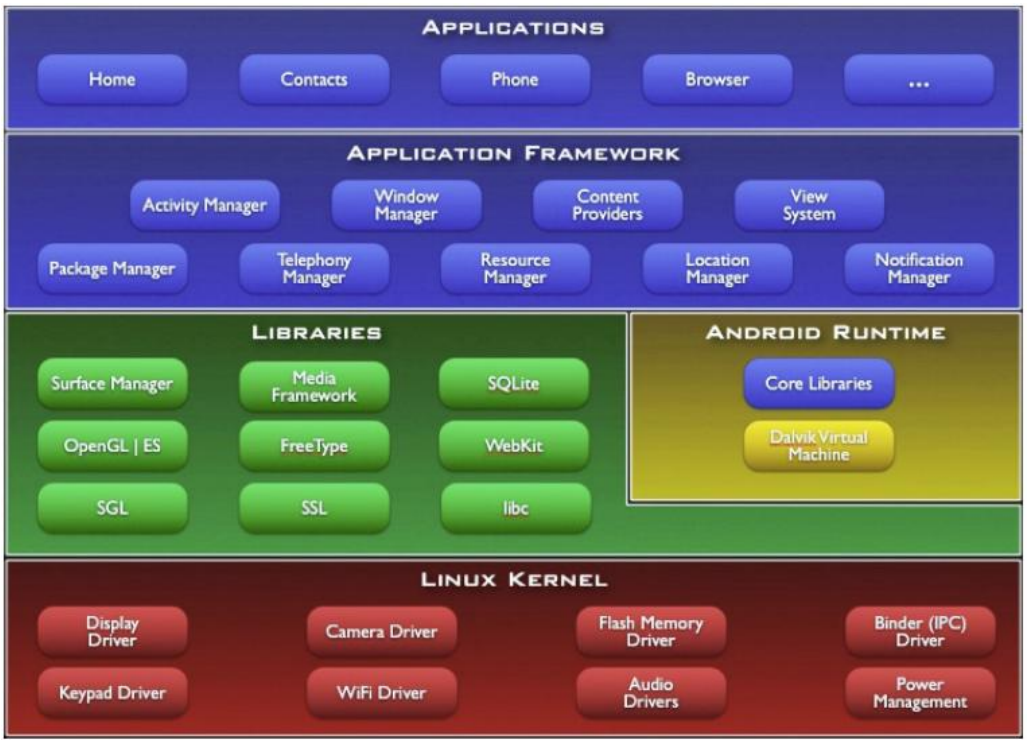

Ilustración 13 Arquitectura del SO Android (Zechner & Green)

*El núcleo [kernel] (**Ilustración 14**) es el conjunto de programas que implementa las más primitivas de las funciones del sistema [administración de la memoria, de los procesos y de las entradas y salidas en general] y es la única parte del sistema operativo que reside de manera permanente en la memoria. Las secciones siguientes del sistema operativo se manejan igual que cualquier programa grande; esto es, las páginas de sistema operativo se traen a la memoria sobre demanda, sólo cuando se necesitan, y el espacio de memoria que usan queda libre conforme se llaman otras páginas. UNIX utiliza el algoritmo de remplazo de páginas de menor uso reciente*. (Flynn & McHoes)

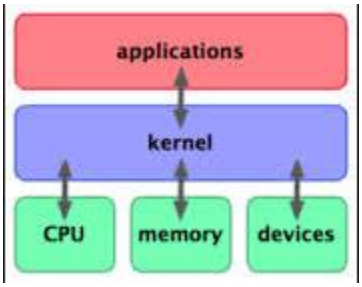

Ilustración 14 Kernel

Si bien pueden ejecutarse comandos de Linux en el sistema operativo Android, los dispositivos móviles no incluyen una consola por default. Se puede acceder a la consola de un dispositivo móvil conectándolo a un equipo de escritorio a través del Android Debug Bridge (Android Open Source Project) . El ABD es distribuido como parte del entorno Eclipse ADT.

ADB es un programa que permite controlar tanto emulaciones como dispositivos, y correr un shell con el propósito de ejecutar comandos en el ambiente de un emulador o un dispositivo (Mednieks & al, 2011).

Los dispositivos Android tienen dos tipos de memoria, la interna y la externa. En la interna se almacena el SO y las aplicaciones. Cuando una aplicación se instala Android crea una carpeta en la memoria interna que será la única localidad a la que tendrá acceso. La memoria Externa es una partición de la memoria del celular que es accesible cuando el dispositivo se conecta a una computadora. Además, un dispositivo puede tener una o dos memorias extraíbles de tipo SD que aumentan su capacidad de almacenamiento.

El lenguaje de programación estadístico R.

De los lenguajes estadísticos más utilizados, R es el único disponible para dispositivos móviles. R es un conjunto integrado de servicios de software para la manipulación de datos, cálculo y representación gráfica. Entre otras cosas, tiene

- un manejo eficaz de los datos y almacenamiento,
- una suite de operadores para cálculos en arreglos y matrices.
- una gran colección de herramientas intermedias para el análisis de datos.
- Herramientas para el análisis y visualización de datos, ya sea directamente en el ordenador o en copia impresa.

Oracle ofrece una versión de R optimizada para el monitoreo en tiempo real de información en una base de datos, incluso de big data (**Ilustración 15**). La versión de Oracle carga dinámicamente bibliotecas de algebra linear del kernel de bibliotecas matemáticas de Intel

(MKL) y AMD (ACML) que activan rutinas multi-hilo optimizadas para un funcionamiento máximo a nivel de hardware (Oracle, 2013)

Ilustración 15. Logotipo de la distribución de R de Oracle

Como puede observarse R es utilizado en la estadística, inteligencia artificial, minería de datos, tratamiento digital de imágenes, etc. R es además interoperable con C y Java, es gratuito, de código abierto y con versiones específicas para sistemas operativos móviles.

La orquestación de un sistema móvil de interés científico

El primer paso en el diseño de una estrategia de comunicación con sensores es la construcción de un marco de referencia arquitectónica (**Ilustración 17**).

Por su funcionamiento, las aplicaciones de medición se clasifican en tres categorías:

- Monitoreo de los procesos y operaciones
- Control de los procesos y operaciones
- Análisis de ingeniería experimental.

*El **monitoreo de los procesos y operaciones** se da en situaciones en donde los aparatos de medición se utilizan para medir cantidades. Sencillamente indican la condición del ambiente y su lectura nos sirve para funciones de control en el sentido normalmente entendido.*

*El **control de los procesos y operaciones** es una de las más importantes aplicaciones de la medición. Esto suele referirse a un sistema automático de control por retroalimentación (**Ilustración 16**).*

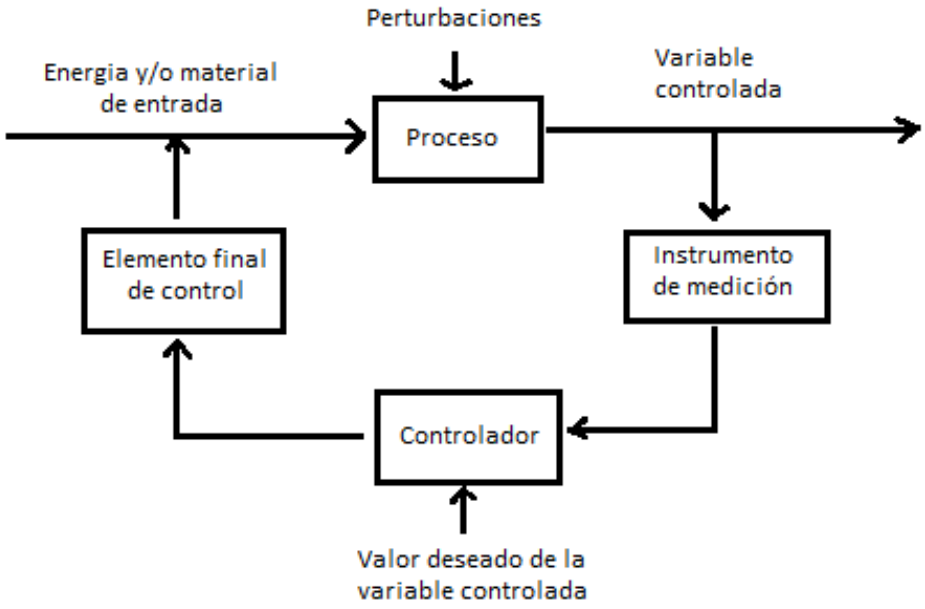

Ilustración 16 Sistema de control por retroalimentación

*El **análisis de ingeniería experimental** es la parte del diseño, desarrollo e investigación de ingeniería que se apoya en pruebas de laboratorio de una clase u otra para dar respuesta a preguntas.* (Doebelin, 2005)

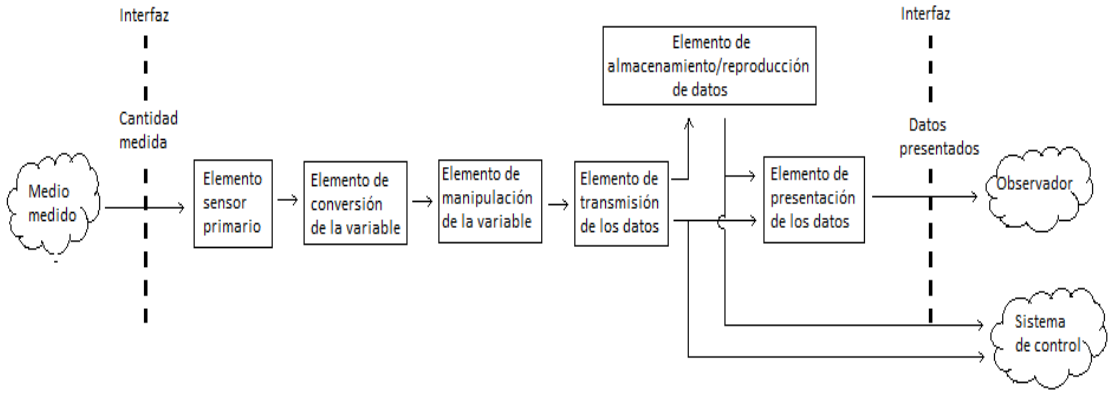

Ilustración 17 Elementos funcionales de un sistema de medición

Al desarrollarse aplicaciones para dispositivos móviles debe tenerse en cuenta las limitantes de memoria y la amplia gama de dispositivos. El tamaño de las pantallas, las versiones de hardware y SO condicionan el desarrollo.

En el sitio de Android encontraremos el programa Android de compatibilidad. Éste es un documento que es definido en conjunto con las empresas para definir las especificaciones de hardware mínimas utilizadas en cada versión de Android. En el Compability Definition Document (CDD) (Gooogle Inc., 2013) encontramos directrices útiles como:

- Latencia mínima del audio (variable)
- tamaño mínimo de la pantalla (2.5 pulgadas)
- densidad mínima de la pantalla (100 dpi)
- relación de aspecto aceptable (4:3 a 16:9)
- Aceleración de gráficos 3D (OpenGL ES 1.0 requerida) (Zechner & Green)

El desarrollo de una aplicación móvil inicia con el diseño de los aspectos de más alto nivel tales como las interfaces, los controles y sonidos.

El modelado de aplicaciones móviles tendrá en cuenta tres aspectos:

- *Interfaz reducida.*
- *Sistema de navegación simple e intuitiva.*
- *Aprovechamiento de características especiales como el GPS, cámara, acelerómetro.* (Vera, 2013)

Las aplicaciones también pueden guardar información en la base de datos SQLite que viene precargada en todos los dispositivos Android. Cada que se instala una aplicación el S.O. le crea una base de datos a la que solo ella puede acceder.

Materiales y métodos.

Introducción

En esta sección se describe el **hardware** utilizado para el desarrollo de esta investigación. Posteriormente se mencionan las herramientas y requisitos para desarrollar aplicaciones de las siguientes tecnologías.

- **Android**
- **Lenguaje R**
- **Arduino**

Finalmente se explican las consideraciones **metodológicas** asociadas al desarrollo de prototipos de interés científico.

Hardware

Las características de memoria y procesamiento de un dispositivo móvil se vuelven relevantes cuando se ejecutan aplicaciones de interés científico; De su configuración dependen la velocidad de ejecución y la experiencia del usuario.

Para el desarrollo de esta investigación se utilizó una tableta electrónica *Sony Xperia S* corriendo Android 4.1 *Jelly bean* con un procesador *NVIDIA Tegra 3* de cuatro núcleos y 16GB de memoria interna.

Para la programación de las aplicaciones móviles se utilizó una computadora de escritorio con un procesador *AMD Phenom* II X3 y 16GB de RAM que ejecuta *Windows 7*.

La cámara fotográfica de la tableta es de 8 megapixeles, pero en el caso específico de la aplicación de ejemplo desarrollada en esta investigación se necesitaba que las fotografías tomadas fueran siempre de una resolución y relación de aspecto específica, aspectos que establece la aplicación que va a hacer uso de la cámara.

La tableta o celular debe contar con al menos 140 MB de memoria interna disponibles para instalar R.

Para los equipos que ejecutan un S.O. Windows es necesario instalar el driver correspondiente del dispositivo móvil a utilizar, tanto para hacer uso de su consola como para desarrollar una aplicación Android. El driver podría no distribuirlo el fabricante del equipo; Google ofrece un driver genérico con soporte a una amplia gama de celulares y tabletas.

Para el SO Linux no se requiere driver y muchos de los comandos del SO son útiles al momento de realizar las aplicaciones por lo que es buena idea utilizar alguna distribución como Ubuntu© para desarrollar las aplicaciones.

Android

Las aplicaciones para Android se programan con el lenguaje ***Java***. Desde su sitio oficial puede descargarse gratuitamente la versión ***ADT*** (Android Open Source Project) del entorno de desarrollo ***eclipse©*** (The Eclipse Foundation, 2014) junto con las herramientas necesarias para realizar y probar las aplicaciones (**Ilustración 18**). También puede instalarse un plug-in a una versión existente del IDE.

Para este trabajo se utilizó la versión Juno del editor eclipse sobre JDK 6; con ella es posible realizar la codificación de las pruebas unitarias y de los distintos módulos que componen una aplicación completa.

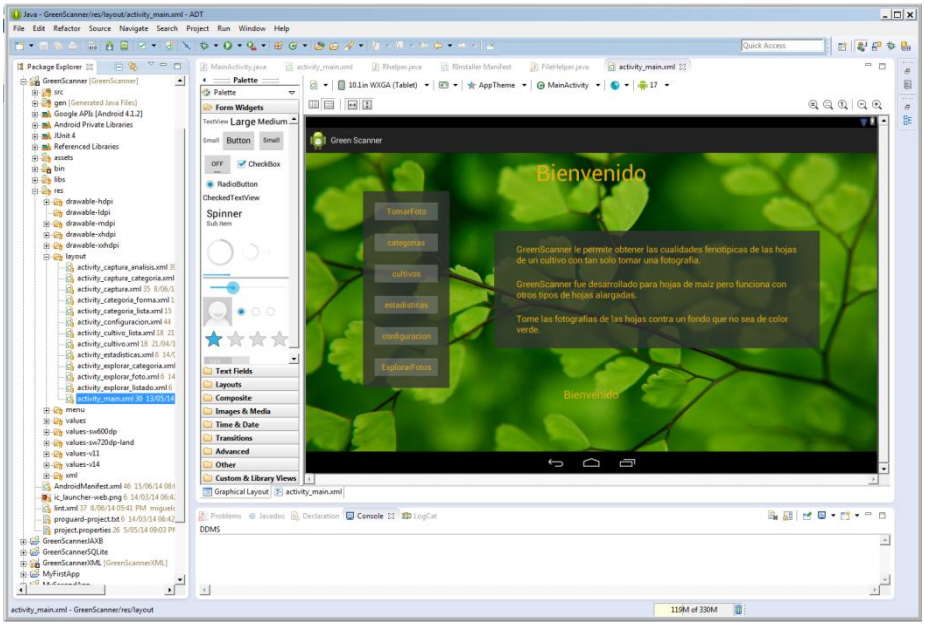

Ilustración 18 El entorno de desarrollo Eclipse

La versión ADT de eclipse incluye algunas herramientas necesarias en el uso del sistema operativo Android. La primera, el SDK manager permite descargar las API's necesarias para desarrollar aplicaciones para las diferentes versiones de Android (**Ilustración 19**).

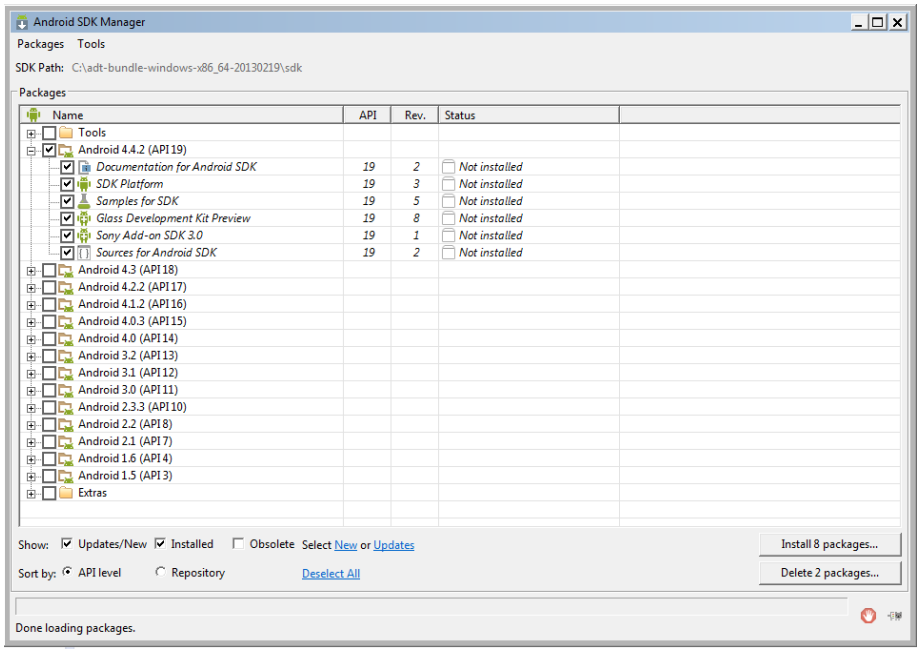

Ilustración 19 Android SDK manager

Otra herramienta es el AVD manager que permite crear máquinas virtuales en las cuales probar las aplicaciones (**ilustración 20**).

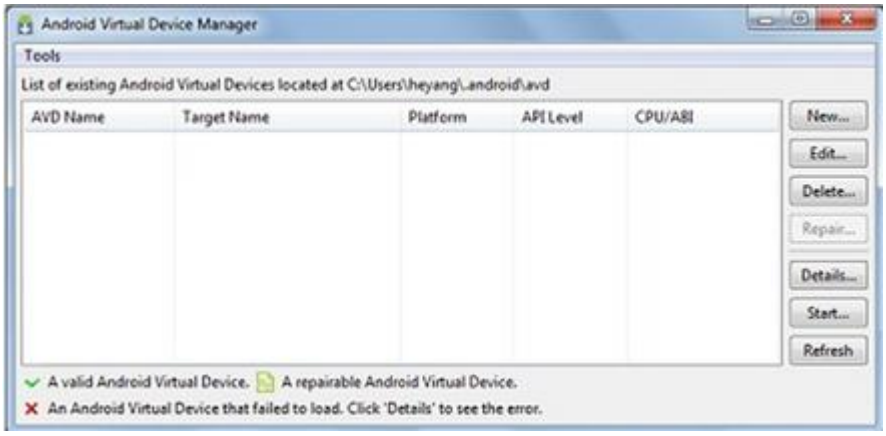

Ilustración 20 Android AVD manager

Lenguaje R

Para desarrollar Scripts del lenguaje R puede utilizarse el entorno de desarrollo CRAN (Comprehensive R Archive Network) que es distribuido gratuitamente desde el sitio oficial de R. Para este trabajo se utilizó la versión 2.15.1 (**Ilustración 21**).

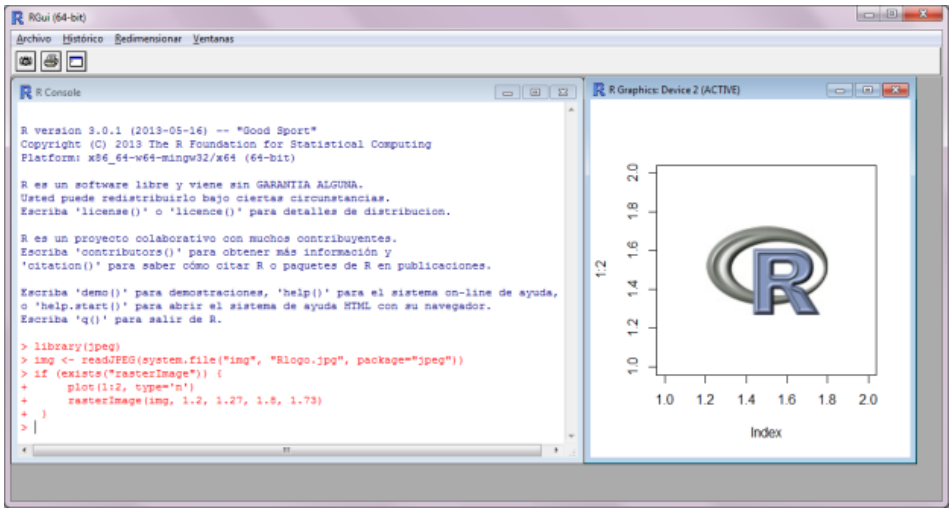

Ilustración 21 El Entorno de Desarrollo CRAN

El editor CRAN debe su nombre a que hay alrededor del mundo un conjunto de repositorios en los que se encuentran alojados las diferentes versiones del editor y de los paquetes que

crea la comunidad. Los paquetes son instalados y actualizados fácilmente mediante comandos.

Existen varios editores comerciales como alternativa a CRAN, un ejemplo es RStudio (RStudio) que presenta la ventaja de mostrar en una ventana adicional, el estado que guardan las variables. Otra opción es instalar un complemento para el editor eclipse.

El editor CRAN sólo incluye un conjunto de paquetes base. Para algunas tareas, como por ejemplo, adquirir una imagen y representarla como matriz numérica, se necesita instalar paquetes adicionales. En los repositorios de CRAN se encuentran disponibles 5175 paquetes.

Arduino

Para la conexión de sensores a un dispositivo móvil se utiliza una tableta ***Arduino©*** (Arduino)***,*** la versión "ADK" es la que puede conectarse a dispositivos móviles.

Una tableta Arduino permite enviar y recibir señales eléctricas de control a través de sus 16 entradas analógicas y 54 digitales; se conecta desde el puerto de datos del celular a un puerto USB del dispositivo electrónico (**ilustración 22**).

Ilustración 22 Tableta Arduino mega ADK rev 2

La programación de la tableta Arduino se desarrolla con el lenguaje C en su editor homónimo distribuido también gratuitamente. En esta ocasión se utilizó la versión 1.0.5 del editor.

El entorno de desarrollo Funciona en *Windows*, *Mac-OS* x y *Linux*. Fue escrito en Java y está basado en *Processing*, *AVR* y otros programas de código abierto.

La instalación del editor incluye la instalación del driver de las tabletas. Las tabletas Arduino que se conectan al puerto USB de la computadora aparecen en el administrador de dispositivos en la sección COM de los puertos seriales (**Ilustración 23**).

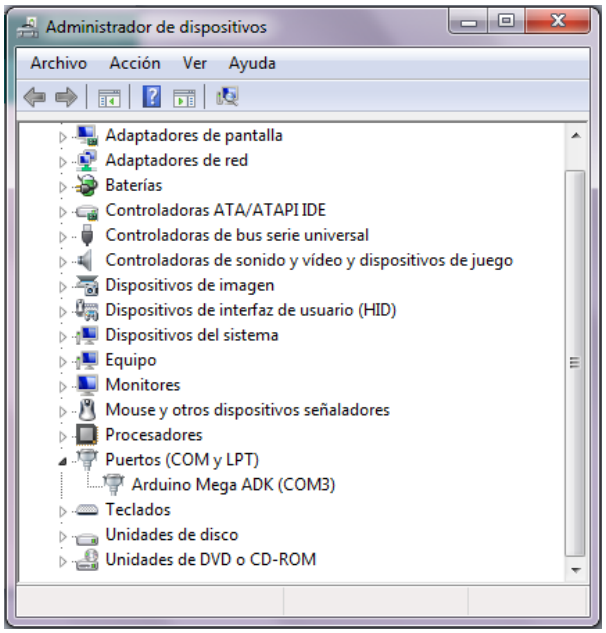

Ilustración 23. La tableta Arduino ocupando el puerto COM 3

En la **Ilustración 24** se muestra el editor de Arduino. En la parte central se halla un script abierto, debajo, de color negro, está la consola. El icono circular con la paloma permite compilar el programa y la flecha que apunta a la derecha carga el programa en la memoria de la tableta.

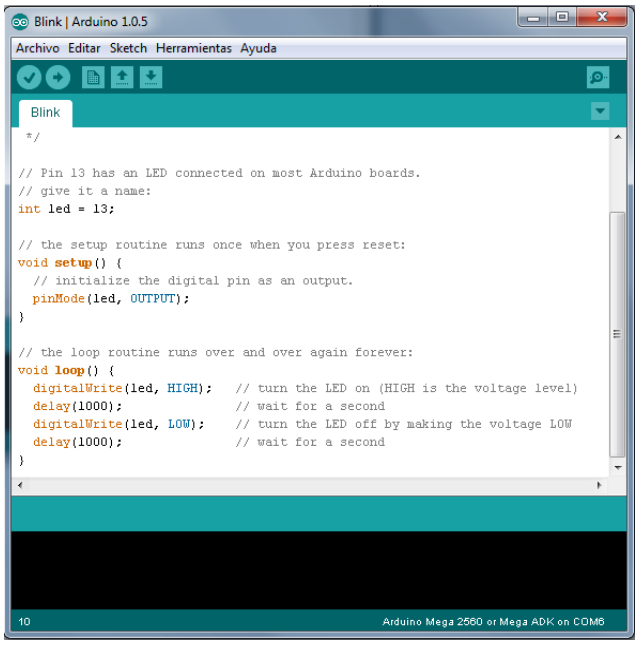

Ilustración 24. El entorno de desarrollo Arduino

La conexión de sensores en Android con Arduino es incluso más sencilla que para equipos de escritorio, pues las API's son parte del SDK.

Metodología

El proyecto desarrollado con la estructura propuesta se divide en módulos diferenciados que pueden desarrollarse de manera independiente aplicando técnicas de desarrollo multicapa.

- Para dispositivos móviles que ejecutan el S.O. Android Las interfaces graficas se desarrollan con Java.
- Para equipos de escritorio la interface gráfica puede desarrollarse con Java.
- En ambos casos, la tableta Arduino a la que se conectan los sensores es programada con C.
- Para equipos de escritorio puede utilizarse el API JRI para ejecutar sentencias de R.
- El API no está disponible para dispositivos móviles pero la clase Process de Java puede ejecutar comandos de R o de cualquier otro lenguaje en una sesión de consola.
- Tanto equipos de escritorio como dispositivos portátiles pueden utilizar la interface R CMD BATCH para ejecutar scripts completos de R

Las diferentes secciones requieren el refinamiento iterativo de su funcionalidad y velocidad de ejecución, sobre todo en la estrategia de procesamiento estadístico. Puede por tanto implementarse una metodología ágil de desarrollo incremental.

El módulo de análisis estadístico puede desarrollarse para equipos de escritorio y trasladarse a dispositivos móviles de forma casi transparente. Los scripts de R pueden tomarse de desarrollos previos sin tener que desarrollar una versión en Java de dichas funcionalidades. Las **ilustraciones 25 y 26** muestran la secuencia de esta migración.

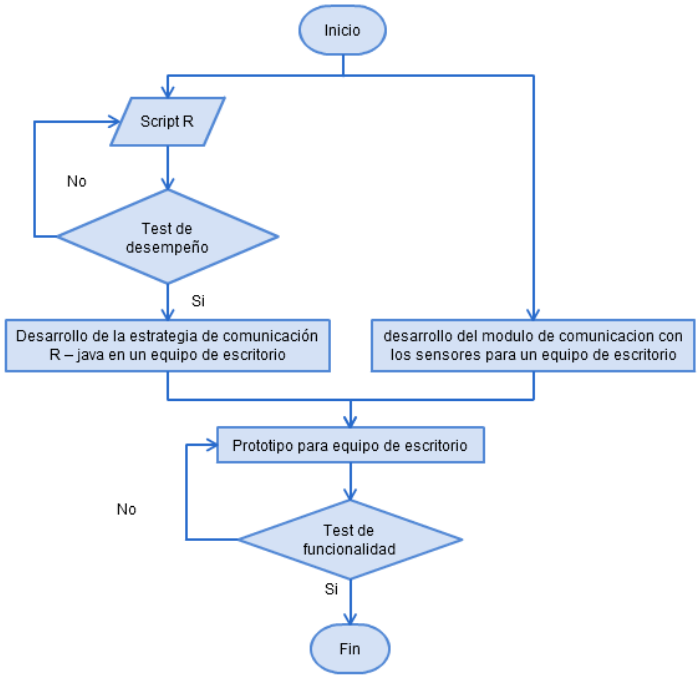

Ilustración 25. Etapas del desarrollo de una aplicación para equipos de escritorio

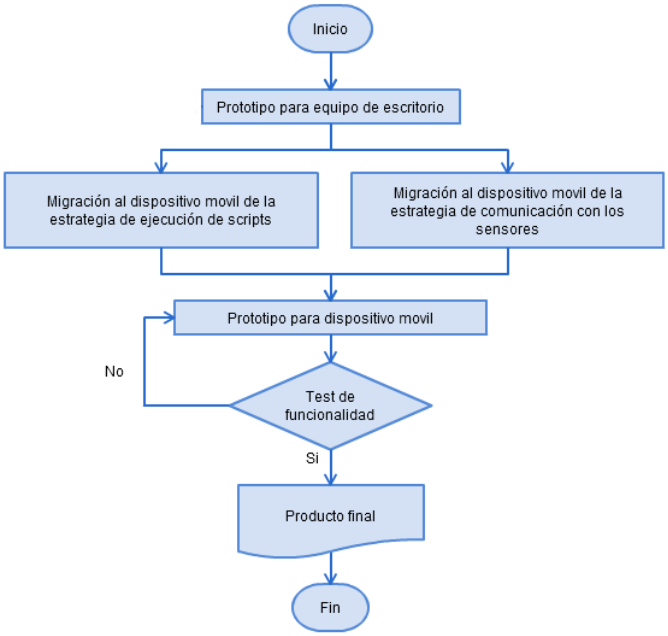

Ilustración 26. Etapas del desarrollo de la versión móvil de la aplicación

La W3C recomienda buenas prácticas que deben observarse en el modelado de aplicaciones móviles (W3C, 2010), estas son:

Dar libertad al usuario:

- Asegurarse de que se informa al usuario el uso que se va a hacer de su información personal y la de su dispositivo.
- Permitir que la aplicación recuerde los datos del usuario de forma automática.
- Permitir que los usuarios puedan elegir la interfaz.

Optimizar el tiempo de respuesta:

- Minimizar el tiempo de espera percibido por el usuario
- Optimizar el tiempo de carga de la aplicación.

Diseñar aplicaciones flexibles:

- diseñar teniendo en cuenta los distintos métodos de interacción existentes
- Asegurarse de que el texto se adapte a la pantalla
- detectar las características del dispositivo

Economizar el uso de la red:

- Comprimir los archivos que se vallan a transferir.
- Dentro de lo posible, evitar utilizar recursos externos.
- Reducir el número de peticiones

Aprovechar las características de los móviles:

- utilizar tecnologías apropiadas para el almacenamiento de datos locales

Las aplicaciones nativas son las que ofrecen una mejor experiencia de usuario. Son aquellas que están especialmente diseñadas e implementadas para el contexto de ejecución. (Ramírez Vique)

1) prerrequisitos

1. entornos de desarrollo

2) implementación

1. Separación de presentación y lógica
2. Modularización para mejorar la compatibilidad

3) Pruebas

1. pruebas unitarias
2. pruebas de estrés
3. pruebas de aceptación

4) firma y distribución

1. licenciamiento y mercados

Para el desarrollo de este trabajo se optó por una metodología ágil. Los estereotipos UML a utilizar en una aplicación móvil son los mismos que para otros tipos de aplicaciones.

El desarrollo ágil es un modelo de desarrollo basado en iteraciones, donde en cada iteración se realizan todas las fases del ciclo de desarrollo.

El desarrollo ágil se basa en los principios del manifiesto ágil y sus valores éticos.

- *Dar más valor a los individuos y a sus interacciones que a los procesos y herramientas.*
- *Dar más valor al software que funciona que a la documentación exhaustiva.*
- *Dar más valor a la colaboración con el cliente que a la negociación contractual.*
- *Dar más valor a la respuesta al cambio que al seguimiento de un plan.*

Con estos valores se intenta conseguir, entre otras cosas, entregar algo lo más pronto posible y evitar problemas originados por cambios de requisitos.

Los métodos ágiles suelen ser muy adecuados para el desarrollo de aplicaciones móviles por las siguientes razones:

- *Alta volatilidad del entorno: Con cambios en entornos de desarrollo, nuevos terminales y nuevas tecnologías a un ritmo mucho más elevado que en otros entornos de desarrollo.*
- *Equipos de desarrollo pequeños: Dado que los desarrollos móviles suelen ser proyectos relativamente pequeños, los equipos no suelen ser muy grandes. Generalmente son llevados a cabo por desarrolladores individuales por PYMEs.*
- *Software no crítico: No suelen ser aplicaciones de alto nivel de criticidad, dado que suelen ser aplicaciones para entretenimiento o gestión empresarial no crítica.*
- *Ciclos de desarrollo cortos: Dada la evolución constante de la industria, se requieren ciclos de vida realmente cortos para poder dar salida a las aplicaciones a tiempo.* (Ramírez Vique)

Capítulo1: Plataforma propuesta

Estructura de la plataforma.

El sistema operativo Android, la plataforma Arduino y el lenguaje R permiten crear ágilmente prototipos de interés científico. La adquisición de lecturas desde sensores puede realizarse mediante una interconexión directa entre los periféricos y el celular o tableta y la lógica de negocios puede realizarse con R **(Ilustración 27).**

Ilustración 27. Objetivos de la plataforma

La interoperabilidad del sistema operativo Android, el lenguaje R y la plataforma Arduino crea nuevos campos de uso para las aplicaciones móviles. La plataforma puede tener aplicaciones de cómputo científico, de estadística y matemáticas o incluso inteligencia artificial.

Otra ventaja de codificar la lógica de negocios y la interface gráfica en lenguajes diferentes (R y Java) es que se favorece la integración de equipos multidisciplinarios. El lenguaje R es mayormente usado en el ámbito estadístico en el que no necesariamente se utilizan lenguajes orientados a objetos.

El uso de Java, R y Arduino es en muchos aspectos similar para equipos de escritorio y móviles. Esto posibilita la creación de versiones móviles de programas desarrollados para equipos de escritorio.

La siguiente tabla muestra una comparativa del uso de R y Arduino en dispositivos móviles y de escritorio. R provee el API JRI para ejecución de instrucciones desde clases de Java.

	Equipos de escritorio	Dispositivos móviles
Conexión de sensores con la plataforma Arduino	Si	Si
Instalación nativa de R	Si	Si
Instalación nativa de CRAN	Si	No
Lenguaje de programación	Java	Java
Ejecución de comandos de R desde java con el API JRI	Si	No
Ejecución de Scripts de R con comandos del S.O.	Si	Si
Variables de entorno	Permanentes	Temporales
Entorno de desarrollo	eclipse	eclipse

Comparativa de las capacidades de equipos de escritorio y móviles

La principal razón por las que no hay una versión de JRI disponible para Android es porque tampoco hay una versión oficial de R para Android. R puede instalarse en Android porque es de código Abierto y puede compilarse para cualquier plataforma basada en Linux, mientras que el código fuente de JRI no está disponible para el público en general. El API JRI utiliza a su vez una biblioteca de vínculos dinámicos (bytecode precompilado) que sin su código fuente es imposible emular.

Otra de las razones por la que, en general, las API's desarrolladas para plataformas de escritorio podrían no funcionar en dispositivos móviles es que sus máquinas virtuales son diferentes y por tanto su compilador.

Una tercera razón es que el API requiere variables de entorno que le indiquen la ruta de instalación de R. Las variables de entorno establecidas en Android con comandos del SO mediante una sesión de consola son temporales, es decir, desaparecerán cuando la sesión se cierre o el dispositivo se apague. Pueden establecerse variables de entorno permanentes modificando el archivo *mkshrc* del SO pero su ubicación varía en cada marca de dispositivo y

el proceso para establecerlas requiere pasos técnicos muy específicos que difícilmente podrían ser realizados por el usuario final. Finalmente, cuando el dispositivo reciba una actualización del SO desaparecerán los cambios realizados en este archivo.

Aunque el API JRI no puede utilizarse (por el momento) en dispositivos móviles, sí pueden ejecutarse comandos de R mediante instrucciones del SO, y pueden ejecutarse instrucciones del SO con emulando el funcionamiento del API.

Un script (conjunto de comandos) del lenguaje R desarrollado en un equipo de escritorio seleccionado para su implementación en un dispositivo móvil debe refinarse iterativamente hasta que su velocidad de ejecución sea la deseada. Debe considerarse que la configuración de hardware presente en un celular o tableta definirá la velocidad de ejecución.

La conexión directa de sensores a un dispositivo móvil haciendo uso de la tableta Arduino disminuye el tiempo destinado a capturar la información al mismo tiempo que asegura la exactitud de las lecturas. La versión ADK r2 de Arduino es la interfaz de facto para el desarrollo de hardware compatible. La tableta puede incluso construirse por el usuario porque es de código abierto.

Al igual que un script de R, el uso de la interface Arduino puede trasladarse de forma casi transparente de su implementación en un equipo de escritorio a un celular o tableta.

Algunas de las bibliotecas definidas para equipos de escritorio no funcionarán en Android, porque la memoria disponible en dispositivos móviles es menor y eso condiciona algunas tareas del sistema operativo. Por ejemplo, el análisis de archivos ***XML*** (eXtensible Markup Language) en equipos de escritorio puede desarrollarse con una estrategia **DOM** (Document Object Model) en la que la totalidad de un archivo XML es leído y cargado en memoria en una estructura tipo árbol. En dispositivos móviles es más apropiado un acceso secuencial como ***SAX*** (Simple API for XML) que no carga los objetos en memoria sino que define un cursor con una posición especifica en el archivo.

Esto es relevante porque encontraremos que los frameworks de persistencia Objeto-relacionales como Hibernate o JPA no están disponibles para dispositivos móviles así que

tendrá que diseñarse una implementación propia. En el caso de intentar almacenar la información en estructuras XML tampoco hallaremos API's para el marshalling objeto-XML como JAXB.

La realización de una aplicación desarrollada con esta estructura considerará un módulo para el lenguaje R, uno para la plataforma Arduino y otro opcional para la persistencia. Los módulos son utilizados por la interface gráfica (**Ilustración 28**).

Ilustración 28. Módulos de la plataforma

Al interior de cada módulo, una capa de servicios agrupa las funcionalidades que el módulo ofrece. El uso de una capa de servicios es una buena práctica que disminuye el acoplamiento con el módulo de la interfaz gráfica facilitando su desarrollo y su migración a nuevas versiones.

En la medida en que la cohesión de los módulos sea observada, su validación independiente agrega robustez a la plataforma al tiempo que disminuirá el tiempo de desarrollo dedicado a diagnosticar y reparar bugs.

Módulos de la aplicación (**Ilustración 29**):

1.- Interface Gráfica: Las aplicaciones Android se componen de ***Activities*** (Ventanas) y ***Views*** (Paneles). Una ventana puede estar construida a partir de varios paneles. Por cada vista desarrollada mediante archivos XML hay una clase de Java que recibe los eventos de la interface gráfica y contiene las variables presentadas al usuario. También hay una clase Java por cada panel.

2.- Módulo de acceso a R: La lógica de negocios realizada en scripts puede ser utilizada indistintamente en equipos de escritorio o móviles. La ***capa de servicios*** muestra el catálogo de funciones estadísticas soportadas. Una clase Helper ordena la ejecución de los scripts y recupera los resultados.

3.- Módulo de persistencia: El motor de bases de datos SQLite viene precargado en todos los dispositivos Android. En este módulo hay un ***Data Acces Object*** que ejecuta los comandos de SQL por cada tabla de la base de datos. Una sola instrucción de la ***capa de servicios*** puede involucrar a más de un ***DAO***, de este tipo de pasos intermedios y ***Triggers*** se encarga la capa ***facade***.

4.- Módulo de acceso a sensores: La aplicación puede enviar instrucciones a los sensores a través de la ***capa de servicios***, o un evento de los sensores desencadena acciones en la interfa

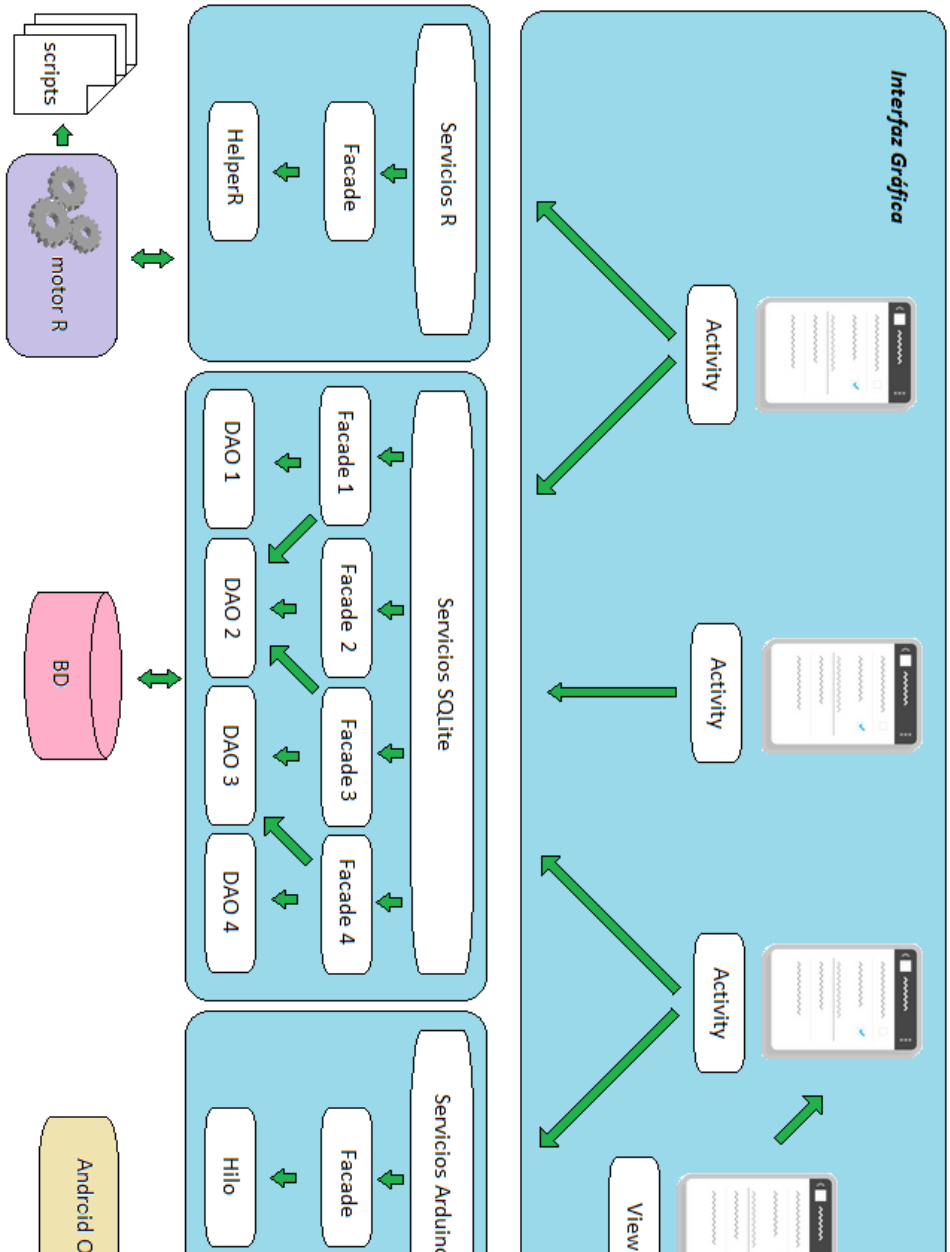

Ilustración 29 Plataforma propuesta

Capítulo 2: La conexión de sensores a un dispositivo móvil.

Integración del framework Arduino

Para desarrollar una aplicación móvil y para utilizar el Framework Arduino es recomendable escoger un dispositivo móvil tan potente y tan común como sea posible. El funcionamiento de las tabletas Arduino está comprobado para las tabletas Nexus de Google pues ejecutan la versión original del S.O. pero también es soportado por la mayoría de dispositivos que ejecutan la versión 3.1 o superior de Android.

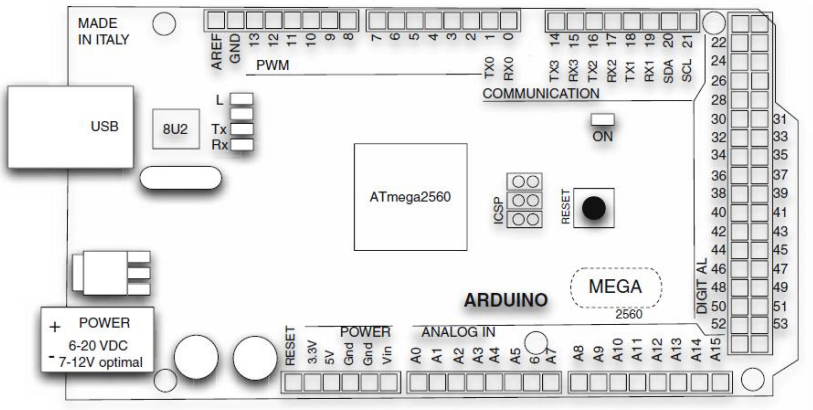

Ilustración 30. Diagrama de la tableta Arduino con sus pines de entrada y salida en las orillas (Evans, 2013)

Las tabletas Arduino trabajan con sensores que funcionan a 3 y 5 volts, tienen 16 entradas analógicas y 54 digitales (**Ilustración 30**). El circuito electrónico de la tableta está disponible de manera abierta en internet, existen además otros proveedores independientes que ofrecen versiones personalizadas de este circuito electrónico.

Como se mencionó en el marco teórico, El dispositivo móvil puede enviar señales de control a la tableta Arduino, o Arduino desencadenar eventos en el dispositivo móvil.

La aplicación móvil *ADK 2012 (**Ilustración 31**)* es un ejemplo de programa que envía señales de control. Dicha Aplicación fue desarrollada por Google como ejemplo para su presentación en 2012 de la siguiente versión del ADK. La aplicación venia acompañada de un reloj configurable desde un celular Android (**Ilustración 32**).

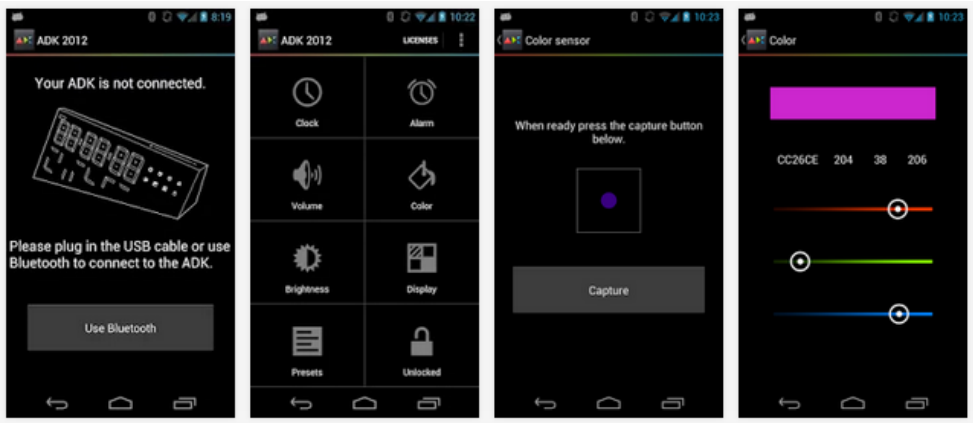

Ilustración 31 ADK 2012

Ilustración 32 Reloj ADK 2012

Para este caso, una activity del módulo GUI envía las instrucciones al módulo de acceso a sensores a través de la capa de servicios. La capa Facade envía a su vez los bytes de la instrucción vía un OutpuStream (**Ilustración 33**).

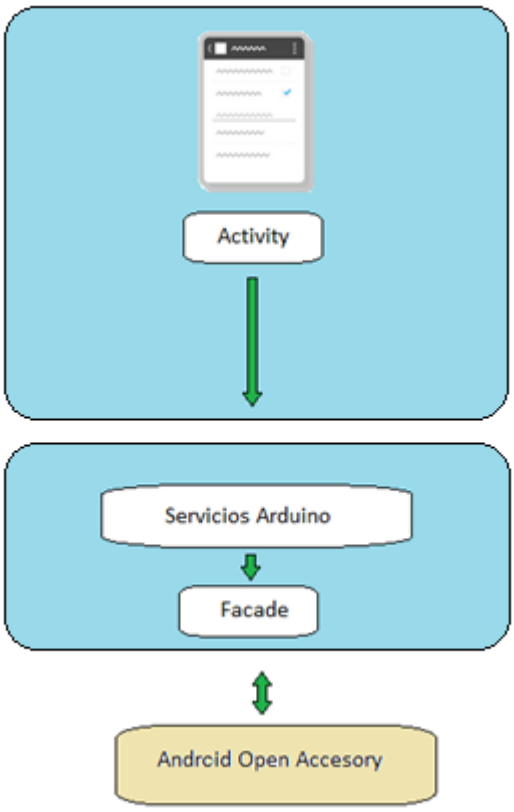

Ilustración 33 Diagrama de módulos para el envió de señales de control

Por otro lado, la **Ilustración 34** muestra una aplicación en la que un potenciómetro (**Ilustración 35**) es conectado a una Tablet usando Arduino. Cuando el potenciómetro aumenta o disminuye su valor de resistencia, un círculo rojo aumenta y disminuye de tamaño. En este ejemplo el componente eléctrico desencadena acciones en la interfaz gráfica.

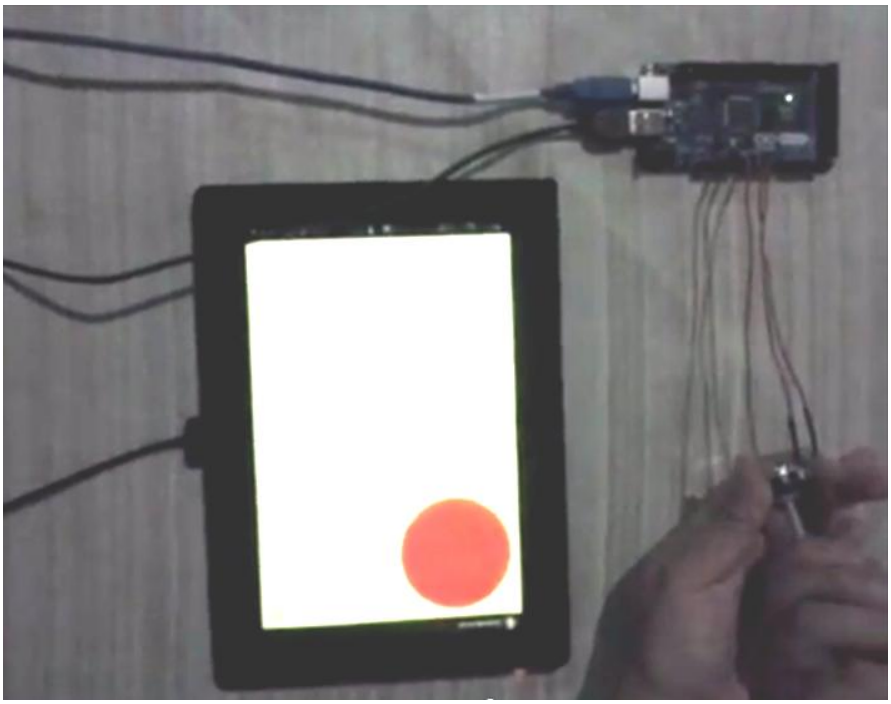

Ilustración 34 Aplicación móvil que utiliza un potenciómetro

Ilustración 35 potenciómetro

En este caso, se utiliza un objeto de tipo Handler que es un hilo que desencadena eventos en una interfaz gráfica. Una Activity (Ventana) se compone de varias View (Paneles) cuyo aspecto es modificado por un Handler (**Ilustración 36**).

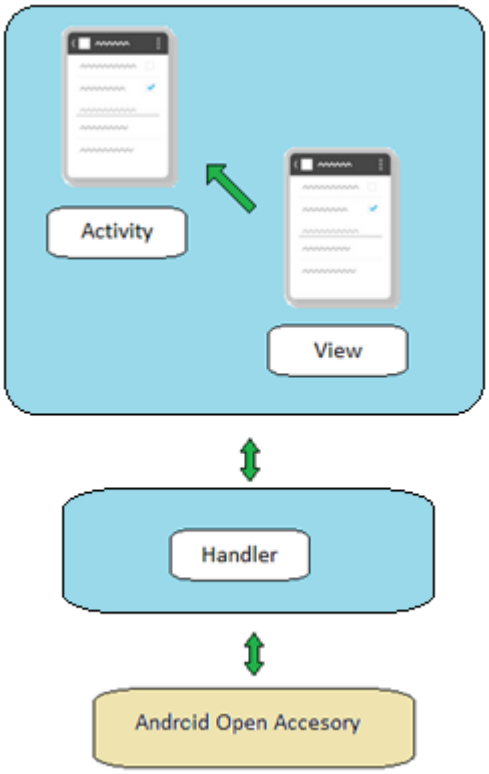

Ilustración 36 Diagrama de módulos para la recepción de señales de control

Scripts Arduino

Las tabletas Arduino se programan mediante scripts con extensión .ino escritos en C++. Los scripts empiezan con la inclusión de bibliotecas, posteriormente se declaran las variables y métodos.

Todos los programas incluyen un método “SETUP” que prepara como puerto de entrada o salida los pines de la tableta.

Finalmente, el método “LOOP” se ejecuta una y otra vez mientras la tableta esté conectada a la corriente eléctrica. Este método revisa con cada ejecución el estado de los sensores y

desencadena las acciones definidas por el programador. Este es el punto en el que se invoca a otros métodos cuando una condición se cumple.

En el **ejemplo 1** en el método setup se establece que el pin 13 sea un puerto de salida de corriente. Un led conectado a la entrada 13 de la tableta Arduino se enciende, transcurre un segundo, se apaga y transcurre otro segundo. Este flujo se realizaría de manera indefinida.

```
int led = 13;
void setup() {
pinMode(led, OUTPUT);
}
void loop() {
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Ejemplo 1. Script Arduino que prende un led intermitentemente.

Los programas al ejecutarse generan una salida al monitor serial **(Ilustración 37)** que es una ventana independiente mostrada en primer plano. En la parte inferior derecha del monitor serial se establecen los Baudios a los que operan los sensores que envían mensajes a la PC.

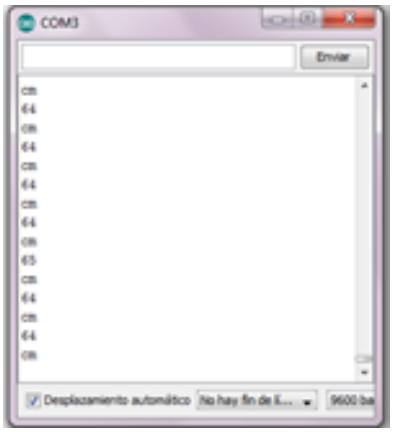

Ilustración 37. Monitor serial

Antes de probar el programa hay que seleccionar en el menú tools/board/ el tipo de tableta que se está utilizando.

En la fase de desarrollo la tableta obtiene energía eléctrica al estar conectada a la computadora, pero también tiene una entrada de alimentación para cuando es usada en producción conectada a un dispositivo móvil.

Para algunos sensores se necesita una biblioteca en lenguaje C que normalmente es provista por el fabricante. Para utilizar una biblioteca basta con incluir los archivos requeridos con extensión .h y .cpp de la biblioteca en la carpeta *libraries* de la ruta de instalación. Si la biblioteca venia acompañarse de ejemplos que aparecerán en el menú *"archivo/ejemplos/"* del editor (**Ilustración 38**).

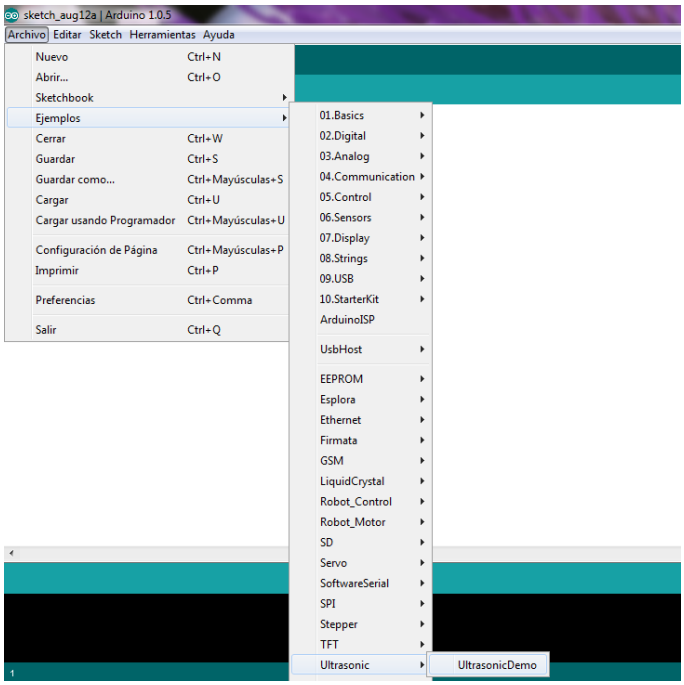

Ilustración 38. Submenú de ejemplos en el IDE Arduino

Configuración del proyecto

Los pasos para configurar un proyecto Android desarrollado con el IDE eclipse son:

- Agregar las Google API's a las propiedades del proyecto.

- Agregar el archivo accesory_filter.xml
- Configurar el archivo descriptor del proyecto (AndroidManifest.xml)

Con el manager SDK debe descargarse el complemento de google API's y agregarlo a las propiedades del proyecto, en la sección Android (**Ilustración 39**).

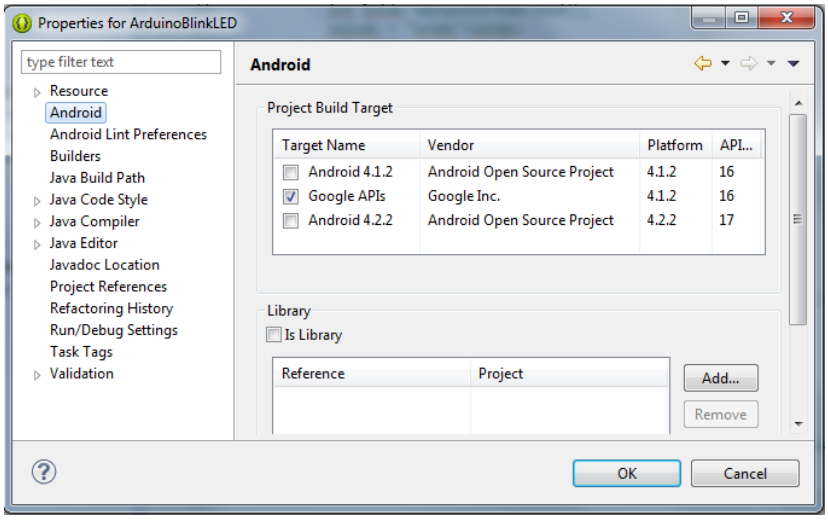

Ilustración 39. Google API's

El **ejemplo 2** muestra el contenido del archivo accesory_filter.xml que debe agregarse en la carpeta res/xml del proyecto. Este archivo es diferente para cada plataforma de conexión de sensores.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <usb-accessory
            manufacturer="Manufacturer"
            model="Model"
            version="1.0" >
    < / usb-accessory >
</resources>
```

Ejemplo 2. Accesory_filter.xml

Todos los programas de android incluyen el archivo AndroidManifest, que es el archivo descriptor de la aplicación, en él, en la parte superior hay que incluir una etiqueta para importar la biblioteca necesaria para interpretar las señales de los sensores (**Ejemplo 3**).

```
<uses-library android:name="com.android.future.usb.accessory" ></uses-library>
```

Ejemplo 3. Etiqueta *Uses-library* del AndroidManifest.xml

También hay que dar de alta los eventos de la conexión del sensor a la tableta, para que en ese momento se inicie la aplicación y al mismo tiempo obtener los permisos necesarios, esto se realiza para la Activity principal. El archivo AndroidManifest.xml completo puede encontrarse en el **anexo B.**

```
<activity
    android:name="mx.uaemex.test.MainActivity"
    android:label="@string/app_name"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name=
            "android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
    </intent-filter>
    <meta-data
        android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
        android:resource="@xml/accessory_filter" >
    </meta-data>
</activity>
```

Ejemplo 4. USB_ACCESSORY_ATTACHED

Implementación

Antes de iniciar la comunicación con sensores hay que definir una clase que represente la unidad de información enviada y recibida. Por ejemplo, para un sensor ultrasónico que mide la distancia a un objetivo la clase quedaría como lo muestra la **Ilustración 40**. La implementación detallada del sensor ultrasónico se abarca en el **capítulo 4.**

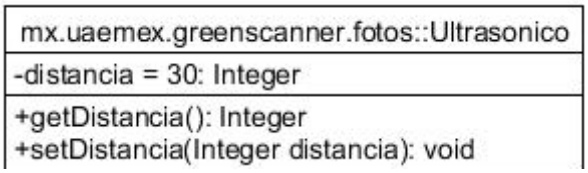

Ilustración 40 Diagrama de la clase Ultrasonico

Como ya se había mencionado antes, una Activity puede estar creada a partir de varios paneles o Views. La **Ilustración 41** muestra el diagrama de clases de un panel que tiene como variable global una instancia de la clase ultrasónico. Este objeto es compartido con la Activity que se habilite para recibir las señales de los sensores. Ya que ambas clases compartirán el objeto ultrasónico, la activity establece el valor de la distancia y la View lo consume para dibujar por ejemplo una barra que crezca y disminuya a la par que cambia el valor.

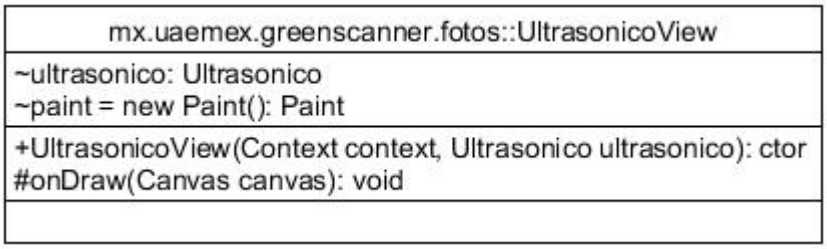

Ilustración 41 Diagrama de Clases de la Clase UltrasonicoView

La Activity posee dos métodos importantes: OpenAccesory y closeAccesory que inician y terminan la comunicación con los sensores. También una variable BroadcastReceiver que representa la tableta Arduino (**Ilustración 42**) ***véase anexo B***.

La Activity se convierte en un hilo al implementar la interface Runnable y sobreescribir el método run que constantemente lee el valor obtenido desde arduino.

El método run y el objeto BroadcastReceiver no necesariamente deben estar en la Activity. Para el ejemplo del **capítulo 4** se decidió utilizarlo de esta manera porque en este caso específico se requiere hacer uso de la cámara.

La toma de la fotografía, la lectura del acelerómetro y la adquisición de las lecturas del sensor ultrasónico se ejecutan en 3 hilos distintos más el hilo de la aplicación y el proceso puede tardar hasta medio segundo. En este caso la Activity orquesta el orden en el que las lecturas

son tomadas y dibujadas y los sensores son prendidos y apagados para evitar que el movimiento de la tableta al tomar la fotografía genere errores.

Ilustración 42 Diagrama de clases del uso de sensores

Cuando hay un valor disponible se desencadenan acciones en la interface gráfica mediante un objeto de tipo Handler. **El ejemplo 5 y 6** muestran un ejemplo en el que un sensor ultrasónico (sonar) es conectado a una tableta. La distancia al objetivo es reflejada en la interface dinámicamente como texto.

```
Handler handler = newHandler() {
        @Override
        publicvoidhandleMessage(Message msg) {
                Integer value = (Integer)msg.obj;
                potenciometro=value;
                textView.setText("Distancia: "+value);
        }
  };
```

Ejemplo 5. Handler

```
@Override
    Public void run() {
    int ret = 0;
    byte[] buffer = new byte[16384];

    while (true) {
            try {
                    ret = mInputStream.read(buffer);
            } catch (IOException e) {
                    break;
            }
            Int i = 0;
            while (i< ret) {
                    int len = ret - i;
                    if (len>= 1) {
                            Message m = Message.obtain(handler);
                            value = (int) buffer[i];
                            m.obj = value;
                            handler.sendMessage(m);
                    }
                    i += 1;
            }
    }
}
```

Ejemplo 6. Método run de la clase MainActivity

La tableta Arduino posee dos entradas, una para la computadora y otra para el celular (**Ilustración 43**).

Ilustración 43. Conexión de la tableta Arduino

Capítulo3: Procesamiento estadístico en un dispositivo móvil

Uso del lenguaje R

El motor de ejecución de R y el entorno de desarrollo CRAN se instalan mediante comandos en los sistemas Linux, y con un archivo ejecutable en para los SO. Windows.

Comandos para instalar R en Linux:

```
sudo apt-get update
sudo apt-get install r-base
```

Ejemplo 7. Comandos para instalar R en Linux

El comando “install.packages(<nombre_paquete>)” instala paquetes extra en una instancia de R. Antes de utilizar una biblioteca instalada es necesario invocarla con el comando “library(nombre_del_paquete)”. Cuando un paquete es descargado, también lo hace su documentación. Los archivos de la ayuda son accesibles mediante el comando “help.start()”.

En el caso de dispositivos móviles, la instalación de paquetes que requieren compilación no se encuentra disponible por default, pero puede conseguirse el código fuente e incluirlo como parte del script.

Los scripts del lenguaje R con extensión .R se ejecutan línea por línea delimitadas las sentencias por punto y coma o por una línea nueva. El símbolo # señala los comentarios y los bloques de código para sentencias de control se delimitan por llaves.

El lenguaje R es sensible a mayúsculas y minúsculas y los nombres de variables aceptan caracteres alfanuméricos, acentos, punto y guion bajo.

Puede utilizarse tanto el signo de igualdad "=" como el de asignación "<-" para otorgarle valores a las variables.

Para denotar un ejemplo de script podemos generar el problema "sumar el contenido de dos vectores de longitud 10, elemento por elemento", la solución es la siguiente:

```
> x <- c(1:10)
> y <- c(11:20)
> z <- x + y
>z
[1] 12 14 16 18 20 22 24 26 28 30
```

Ejemplo 8. Suma de dos vectores

Instalación del lenguaje R en el sistema operativo Android

Existen dos maneras de instalar R en un dispositivo móvil. La primera es instalar la aplicación GNU ROOT en nuestro dispositivo móvil. Con ella se pueden instalar programas como si se tratara de cualquier Linux.

La segunda es obtener permisos de superusuario para instalar R en la carpeta /data/local/gcc.

La distribución del lenguaje R para el sistema operativo Android no posee todas las funcionalidades de las versiones para sistemas operativos de escritorio y el editor CRAN tampoco se encuentra disponible.

La distribución para Android la componen los archivos "android_gcc_supplement.tar.bz2", "android_gcc_r2a.tar.bz2" y "android_R_r1a2.tar.bz2" que en conjunto pesan 170 Megabytes (grosjean, 2013).

No todos los dispositivos móviles cuentan con espacio suficiente para instalar R pero también puede instalarse en una partición de la memoria externa que se monte al sistema de archivos del sistema operativo.

Se puede acceder a la consola de un dispositivo móvil que se conecte a un equipo de escritorio a través del Android Debug Bridge (ABD). El ABD es distribuido como parte del entorno Eclipse ADT.

En un equipo de escritorio al que se ha conectado el dispositivo móvil se inicia una sesión del ADB mediante el comando "*adb Shell"* escrito en la consola.

Los tres archivos de instalación deben copiarse mediante comandos a la carpeta /data/local/gcc del dispositivo para después extraerse. Podría ser necesario software extra para descomprimir los archivos de instalación ".bz2".

Algunos de los comandos requieren ser ejecutados por el usuario root, que en los sistemas Linux es aquel que posee todos los privilegios de lectura, escritura y ejecución. La opción de ejecutar comandos como superusuario en los celulares y tabletas no se encuentra soportada por default;

La responsabilidad de obtener permisos de superusuario es delegada al dueño del dispositivo pues el proceso no ofrece garantías sobre la integridad de los archivos del sistema operativo; además, depende de cada modelo de dispositivo.

La instalación de R en Android se realiza mediante 11 comandos, el primero crea la carpeta de instalación. El comando ***adb push*** copia los archivos de un equipo de escritorio al dispositivo móvil. Bunzip2 y tarxf descomprimen y enlazan las bibliotecas de vínculos dinámicos.

```
mkdir /data/local/gcc
adb push android_gcc_r2a.tar.bz2 /data/local/gcc
adb push android_gcc_supplement.tar.bz2 /data/local/gcc
adb push android_R_r1a2.tar.bz2 /data/local/gcc
cd /data/local/gcc
bunzip2 android_gcc_r2a.tar.bz2
bunzip2 android_gcc_supplement.tar.bz2
bunzip2 android_R_r1a2.tar.bz2
tarxf android_gcc_r2a.tar
tarxf android_gcc_supplement.tar
tarxf android_R_r1a2.tar
```

Ejemplo 9. Comandos para instalar R en Android

Los archivos de la aplicación ya instalada se guardan en el directorio data/local/gcc/bin mientras que las bibliotecas compartidas en data/local/gcc/lib. Al igual que en un equipo de escritorio, es necesario declarar dichas rutas como variables de entorno

```
export LD_LIBRARY_PATH = $LD_LIBRARY_PATH: /data/local/gcc/lib
export PATH=$PATH:/data/local/gcc/bin
```

Ejemplo 10. Variables de entorno

Una vez establecidas las variables de entorno, el comando "R" inicia la aplicación (**Ilustración 44**). Las variables de entorno establecidas por comandos sólo estarán presentes durante la sesión de la consola. Otras opciones para establecer las variables de entorno de tal manera que sean visibles para la aplicación serán discutidas más adelante.

Ilustración 44. Ejecución de R en una tableta electrónica

Instalación en una máquina virtual del sistema operativo Android

El ADT incluye utilidades para la creación de máquinas virtuales, el ADB ejecuta los comandos en la máquina virtual si no encuentra ningún dispositivo físico. El proceso de instalación en una máquina virtual de Android sólo difiere en que no es necesario obtener privilegios de superusuario (**Ilustración 45**).

La opción “savesnapshots” seleccionada durante la creación de la máquina virtual permite guardar su estado entre ejecuciones, de otro modo la instancia retorna a su estado original cada vez que se apaga.

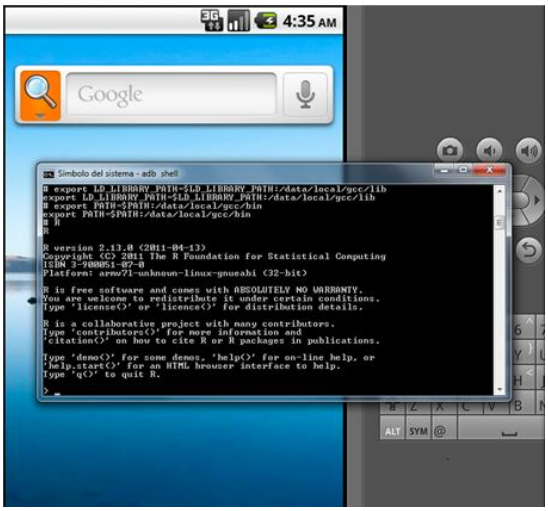

Ilustración 45. Ejecución de R en una máquina virtual

Invocar sentencias del lenguaje R con el API JRI

Una aplicación para equipos de escritorio desarrollada con java puede ejecutar instrucciones del lenguaje R usando el API JRI que se obtiene cuando el paquete rJava es instalado.

El API incluye un archivo de biblioteca compartida con extensión *".DLL (Dinamyc Link Library)"* para sistemas Windows o *".SO (SharedObject)"* para sistemas Linux.

La clase org.rosuda.JRI.Rengine posee el método público *"eval"* que recibe la instrucción de R como parámetro y el método *"end*" que finaliza la sesión.

El API JRI es distribuido dentro del paquete rJava. Es recomendable actualizar los paquetes ya instalados antes de agregar uno nuevo. Será necesario definir las siguientes variables de entorno:

```
R_HOME:C:\Program Files\R
LD_LIBRARY_PATH:C:\Users\user\ Documents\R\win-library\3.0 \rJava\jri
Path:…; C:\Program Files\R\R-3.0.1\bin \x64;
    C:\Users\user\Documents\R\win-library\3.0 \rJava\jri\x64
```

Ejemplo 11. Variables de entorno JRI

El siguiente código Java instancia un objeto de tipo Rengine del cual utiliza el método eval para evaluar una sentencia del lenguaje R que retorna un número entero.

```
Rengine re=new Rengine (new String [] {"--vanilla"}, false, null);
    if (!re.waitForR()) {
            re.end();
    } else {
            Integer result = re.eval (COMANDO).asInteger();
            re.end();
    }
}
```

Ejemplo 12. Ejecución de un comando de R con el API JRI

Variables de entorno

En el sistema operativo Windows las variables de entorno pueden listarse y establecerse con el comando "*SET*". En el sistema Linux las variables se establecen con el comando "*EXPORT*".

Las variables de entorno establecidas vía la consola del ADB sólo están presentes durante la sesión de usuario, es decir, cuando se apaga el dispositivo desaparecen.

Para establecer las variables de entorno permanentemente es necesario modificar el archivo "mkshrc" cuya ubicación varía entre los distintos modelos de dispositivos móviles. Aún con permisos de superusuario esto no es posible pues se trata de un archivo de sólo lectura perteneciente al sistema operativo.

Para modificar este archivo es necesario desvincular del sistema operativo la carpeta *system* para entonces modificar los privilegios de lectura y escritura.

El siguiente comando desmonta la carpeta system para después volverla a montar modificando los permisos de lectura y escritura:

```
Mount –o remount,rw –t yaffs2 /dev/block/mtdblock3
```

Ejemplo 13. remount RW

El archivo puede modificarse entonces con un editor de texto para después retornar la carpeta system a su estado original:

```
Mount –o remount,ro –t yaffs2 /dev/block/mtdblock3
```

Ejemplo 14. remount RO

Modificar un archivo perteneciente al sistema operativo es una solución demasiado invasiva que de hecho se revertirá en cuanto el usuario actualice su sistema operativo.

Las variables de entorno también pueden establecerse por proyecto en el editor eclipse pero esta opción no está disponible para dispositivos móviles.

Las variables de entorno tambien pueden establecerse programaticamente. Una aplicación de java puede establecer las variables de entorno a utilizar a travez de la clase System. Sin embargo esto no funciona para utilizar el API JRI pues la biblioteca distribuida como archivo

jar utiliza a su vez una biblioteca de terminacion .dll que toma las variables de entorno del S.O. Las variables establecidas programaticamente estarán disponibles para la instancia del programa Java pero no pará la biblioteca dinamica del API.

El API JRI es una buena opcion para aplicaciones de escritorio pero no resulta practica en el SO. Android. A continuacion se presenta una opcion disponible para ejecutar scripts.

Invocar la ejecución de un script del lenguaje R desde una aplicación móvil

Una opción disponible tanto para equipos de escritorio como para dispositivos móviles es la ejecución no interactiva de un script mediante líneas de comandos. R provee la interfaz R CMD BATCH que puede ser invocada desde consola.

```
R CMD BATCH [options] infile [outfile]
```

Ejemplo 15. Ejecución no interactiva de archivos por lotes

Donde "*infile"* es el script a ejecutarse con extensión ".R", "*outfile"* es un archivo de texto donde se guardará la salida de la ejecución y "options" es una lista de argumentos que configuran la ejecución del script.

Para utilizar esta opción desde una aplicación diseñada para android puede incorporarse la creación de un objeto del tipo java.lang.Process mediante el cual ejecutar 5 comandos: el primero, "SU", obtiene permisos de superusuario, dos comandos más para establecer las variables de entorno de R y la siguiente para ejecutar el script de R.

```
public void ejecutarScriptR() throws Exception{
Process p = Runtime.getRuntime().exec( "su" );
DataOutputStreamos = new DataOutputStream(p.getOutputStream());
os.writeBytes("export LD_LIBRARY_PATH = $LD_LIBRARY_PATH:/data/local/gcc/lib \n" );
os.writeBytes("export PATH=$PATH:/data/local/gcc/bin \n" );
os.writeBytes( "R CMD BATCH miScript.R outLog.txt\n" );
os.writeBytes("exit\n" );
os.flush();
os.close();
p.waitFor();
}
```

Ejemplo 16. Uso de la interface R CMD BATCH desde una clase Java

Al utilizar esta opción se vuelve necesario que el script de R se diseñe para guardar el resultado de su ejecución en un archivo accesible para la aplicación Android. El **ejemplo *creaProperties*** del **anexo B** muestra la función de R que genera un archivo properties a partir de recibe un arreglo de claves y un arreglo de valores.

Desarrollo de un instalador del lenguaje R

R debe ser instalado en la carpeta /data/local/gcc perteneciente a la memoria interna. Las aplicaciones desarrolladas para Android no pueden modificar archivos de la memoria interna más que de su carpeta de instalación, ni siquiera con permisos de superusuario.

Solo mediante comandos del Shell de linux pueden realizarse estas acciones, las instrucciones pueden ejecutarse con java invocando la ejecución de archivos por lotes (.sh) o mediante la interface Procces de java.

Del mismo modo que eran ejecutadas las instrucciones de R pueden ejecutarse los siguientes comandos necesarios:

- **mv [from] [to]** : mover un archivo de una localidad a otra
- **mkdir [folder]** : crear una carpeta o archivo
- **tar xf [file] -C [path destino]** : descomprimir archivos tar.gz
- **bunzip2 [file]**: descomprimir archivo bz2.

La estrategia para el instalador es:

1. Las aplicaciones Android tienen una carpeta llamada assets en la que se almacenan recursos que la aplicación deba utilizar. en esta carpeta pueden colocarse los tres archivos de R necesarios para la instalación.
2. Pasar los archivos a la memoria externa.
3. Crear la estructura de carpetas necesarias en la memoria interna.
4. Descomprimir los archivos .tar al tiempo que son pasados de la memoria externa a la interna.
5. Descomprimir los archivos bz2.
6. Realizar un test utilizando la interface Procces y estableciendo las variables de entorno.

```
public static void proccesRootCommands(List<String> commands)throws Exception{
        Process p = Runtime.getRuntime().exec( "su" );
        DataOutputStream os = new DataOutputStream(p.getOutputStream());
        for(String command:commands){
                os.writeBytes(command+" \n" );
        }
        os.writeBytes( "exit\n" );
        os.flush();
        os.close();
        p.waitFor();
    }
```

Ejemplo 17. Método para ejecutar comandos linux como superusuario

Los pasos de esta estrategia requieren que el paso anterior se haya concluido antes de realizar el siguiente. Debido a que cada actividad es un hilo independiente se requiere un método que valide que la tarea anterior se haya completado, sobre todo en la creación de archivos. En el ***Anexo B*** el método *validarArchivoExista* realiza un bucle cada n milisegundos hasta que el archivo especificado exista o el tiempo máximo de espera se cumpla.

La aplicación solo presenta una ventana en la que un botón inicia la instalación. Conforme las tareas se completan su check box es activado (**Ilustración 46**).

Ilustración 46. Ejecución del instalador

Capítulo 4: Caso de estudio aplicado

Introducción

En este capítulo se ejemplifica el uso de la plataforma propuesta para migrar una aplicación de escritorio a dispositivos móviles conservando la lógica de negocios. La aplicación para equipos de escritorio que permite obtener el área de las hojas de las plantas desde imágenes obtenidas de un escáner, la versión móvil lo realiza a partir de fotos tomadas con el dispositivo.

El tamaño de las hojas de las plantas es uno de los aspectos más importantes en el estudio de la salud de un cultivo pues está directamente relacionado con su capacidad para absorber la luz solar e intercambiar humedad y CO2 con su ambiente.

Mediante el análisis del área foliar puede medirse el efecto de una plaga sobre una planta, la eficacia de las técnicas y substancias empleadas en su cuidado o cuantificar su volumen de producción.

Las técnicas para determinar el área foliar se dividen en destructivas y no destructivas, dependiendo de si la hoja estudiada es separada o no de la planta. Dichas técnicas intentan contrarrestar los errores presentes en una estimación manual.

Un primer método de medición es la utilización de un integrador electrónico. Existen integradores portátiles y de laboratorio. Estos equipos son escáneres cuya principal desventaja es su costo elevado. La **Ilustración 47** muestra el integrador CL-202L de la marca CID Bio Science cotizado en 5,755 USD. (Forestry Suppliers).

Ilustración 47 Integrador CI-202L

La segunda técnica es la obtención del área de una hoja mediante estimaciones matemáticas con la desventaja de que debe diseñarse un modelo matemático por cada distinto cultivo a estimar. La exactitud de un modelo estadístico depende de la técnica empleada y de la calidad de las muestras utilizadas para su desarrollo.

Las técnicas estadísticas son una alternativa al uso de equipo especializado de medición pero dependen a su vez de otras mediciones que, de realizarse manualmente, pueden propagar errores.

En cuanto a las técnicas destructivas, la medición en condiciones de laboratorio se centra en el uso de aplicaciones para el tratamiento digital de imágenes. Un ejemplo es la aplicación ImageJ® que es una herramienta de propósito general que permite aplicar transformaciones a imágenes digitales.

La aplicación de tratamiento digital de imágenes presentada a continuación es específica para obtener el área foliar por lo que el usuario sólo requiere seleccionar la imagen a analizar.

Otras opciones requieren que el usuario aumente el contraste, elimine el ruido, aplique filtros, etc. antes de obtener el área en pixeles. Además, debe instrumentar una técnica para convertir de pixeles a cm en imágenes tomadas, por ejemplo, con cámaras digitales de una resolución y a una distancia variables.

La interface gráfica de la aplicación fue desarrollada con lenguaje java en el IDE Netbeans y el tratamiento digital de las imágenes se realizó mediante la ejecución de un script del lenguaje R.

Posteriormente se presenta una versión para dispositivos móviles en el que la lógica de negocios codificada con R es mantenida, pero ahora se emplea para analizar fotografías tomadas con el celular o tableta. La plataforma Arduino es utilizada para conectarle un sensor ultrasónico al dispositivo móvil de tal manera que se conozca la distancia a la que fue tomada la foto.

Aplicación para obtener el área foliar desde imágenes obtenidas de un escáner

Introducción

A continuación se presenta el desarrollo de una aplicación para equipos de escritorio que permite obtener el área de hojas simples alargadas a partir de imágenes digitales obtenidas mediante un scanner convencional.

La interface gráfica está desarrollada con Java y el tratamiento digital de las imágenes fue desarrollado con el lenguaje estadístico R.

Materiales, métodos e implementación.

La aplicación realiza el tratamiento digital de las imágenes mediante la ejecución de un script del lenguaje R en una instancia instalada por el usuario.

La interface gráfica está desarrollada con lenguaje Java. El desarrollo en diferentes plataformas de la lógica de negocios y la interface gráfica permite que ambas secciones sean codificadas paralelamente ahorrando tiempo y favoreciendo la integración de equipos multidisciplinarios **(Ilustración 48).**

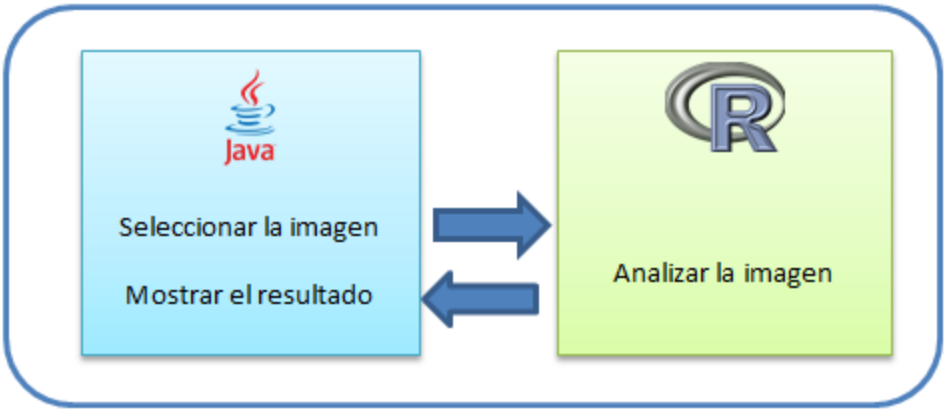

Ilustración 48. Módulos de la aplicación

El programa fue desarrollo para el sistema operativo Windows© pero puede utilizarse en Linux© y Macintosh© pues tanto R como Java son multiplataforma.

La aplicación presentada se desarrolló en la versión 8 del entorno de desarrollo gratuito NetBeans. Por su parte R distribuye, también gratuitamente, el entorno de desarrollo CRAN (Comprehensive R Archive Network).

La ubicación en la que sea instalado R debe publicarse como variables de entorno de tal manera que el programa de Java puede establecer la comunicación **(Figura 2)**. Las variables de entorno son una lista de claves que el sistema operativo almacena sobre la ubicación de archivos y programas.

- R_HOME: C:\Program Files\R\R-2.15.1
- Path: …;C:\Program Files\R\R-2.15.1\bin\x64;

Ejemplo 18. Variables de entorno

El script de R que analiza las imágenes digitales requiere de la instalación de 3 de estos paquetes.

Paquete JPG:

Este paquete proporciona una manera fácil y sencilla de leer, escribir y mostrar imágenes de mapa de bits almacenados en el formato JPEG. Puede leer y escribir vectores en memoria o hacia archivos (jpeg, 2014).

Paquete BMP

Lee imágenes en formato BMP de Windows. Actualmente se limita a imágenes en escala de grises a 8 bits e imágenes (A) RGB de 24 y 32 bits. Aplicación R sin dependencias externas (bmp, 2014).

Paquete PIXMAP

Funciones para la importación, exportación, impresión y demás manipulaciones de imágenes de mapas de bits (pixmap, 2014).

Para instalar paquetes en el sistema R tan solo hay que iniciar una sesión en el editor CRAN y ejecutar el comando *install.packages('nombre_del_paquete').*

La aplicación supone que el usuario digitalice sus muestras como imágenes en formato jpg con la ayuda de un escáner convencional **(Ilustración 49)** teniendo en cuenta que la resolución y tamaño de las imágenes tienen una incidencia directa en la velocidad de ejecución del análisis.

Ilustración 49. Digitalización de las hojas

Las imágenes en formato JPG obtenidas desde un scanner tienen como parte de sus metadatos su densidad, es decir, el número de pixeles que representan una pulgada cuadrada. Este valor es utilizado por las impresoras para imprimir las imágenes en su tamaño correcto. También es utilizado por una computadora para desplegar las imágenes en un monitor en su tamaño real y es utilizado por la aplicación para convertir los resultados de pixeles a centímetros.

Conversión:

- cm= pixeles*2.54/ppp

La **Ilustración 50** muestra las propiedades de una imagen JPG a las que se aceden con click derecho/ propiedades en Windows.

La **Ilustración 51** muestra un fragmento de los metadatos de una imagen en formato JPG en la que las propiedades "*numLines*" y "*samplesPerLine*" de la etiqueta "*sof*" hacen referencia al alto y ancho de la imagen en pixeles. Por su parte, las propiedades "*Xdensity*" e "*Ydensity*" de la etiqueta "*app0JFIF*" muestran los pixeles que, vertical y horizontalmente, dibujan una pulgada cuadrada en su escala original.

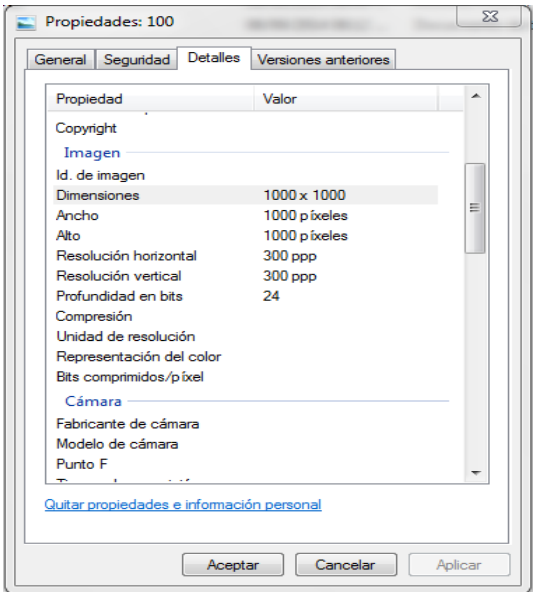

Ilustración 50 Propiedades de una imagen jpg

```
<javax_imageio_jpeg_image_1.0>
    <JPEGvariety>
        <app0JFIF
            majorVersion="1"
            minorVersion="1"
            resUnits="1"
            Xdensity="96"
            Ydensity="96"
            thumbWidth="0"
            thumbHeight="0"/>
    </JPEGvariety>
    <markerSequence>
        <dqt></dqt>
        <sof
            process="0"
            samplePrecision="8"
            numLines="513"
            samplesPerLine="931"
            numFrameComponents="3">
        </sof>
        <dht></dht>
        <sos></sos>
    </markerSequence>
</javax_imageio_jpeg_image_1.0>
```

- **Ilustración 51. Parte de los metadatos de una imagen en formato JPG**

La clase javax.imageio.ImageIO permite tener acceso a los metadatos de una imagen desde un programa de java. La estructura XML de los metadatos de las imágenes es diferente para cada tipo de archivo. Actualmente la aplicación solo soporta el formato JPG.

La estrategia de funcionamiento de la aplicación es la siguiente **(Ilustración 52)**:

1.- El usuario selecciona la imagen a analizar.

2.- La aplicación ejecuta un script del lenguaje R para analizar la imagen seleccionada.

3.- R genera un archivo de texto con el mismo nombre y ubicación que la imagen pero de tipo *properties* en el que se escriben el área, ancho y alto de la hoja en pixeles y centímetros además de las coordenadas de la base y de la punta de la hoja.

4.- R genera un archivo de imagen con el mismo nombre y ubicación que la imagen original pero en formato de mapa de bits de 256 colores que contiene la versión tratada digitalmente.

5.- la aplicación despliega tanto la información del archivo properties como la imagen tratada digitalmente.

```
width_px = #
high_px = #
area_px = #
width_cm = #
high_cm = #
area_cm = #
izq_x = #
izq_y = #
der_x = #
der_y = #
```

Ejemplo 19: Campos presentes en el archivo de propiedades.

Gracias a que los resultados son almacenados en el lugar donde se encuentre la imagen a analizar, es el usuario el que establece la estructura de carpetas para organizar sus archivos.

Cuando el usuario selecciona una imagen que ya fue analizada entonces solo despliega los resultados definidos en el archivo de propiedades de la imagen.

El contenido del archivo properties y la imagen resultante del análisis pueden ser revisados posteriormente por el usuario, incluso sin utilizar esta herramienta.

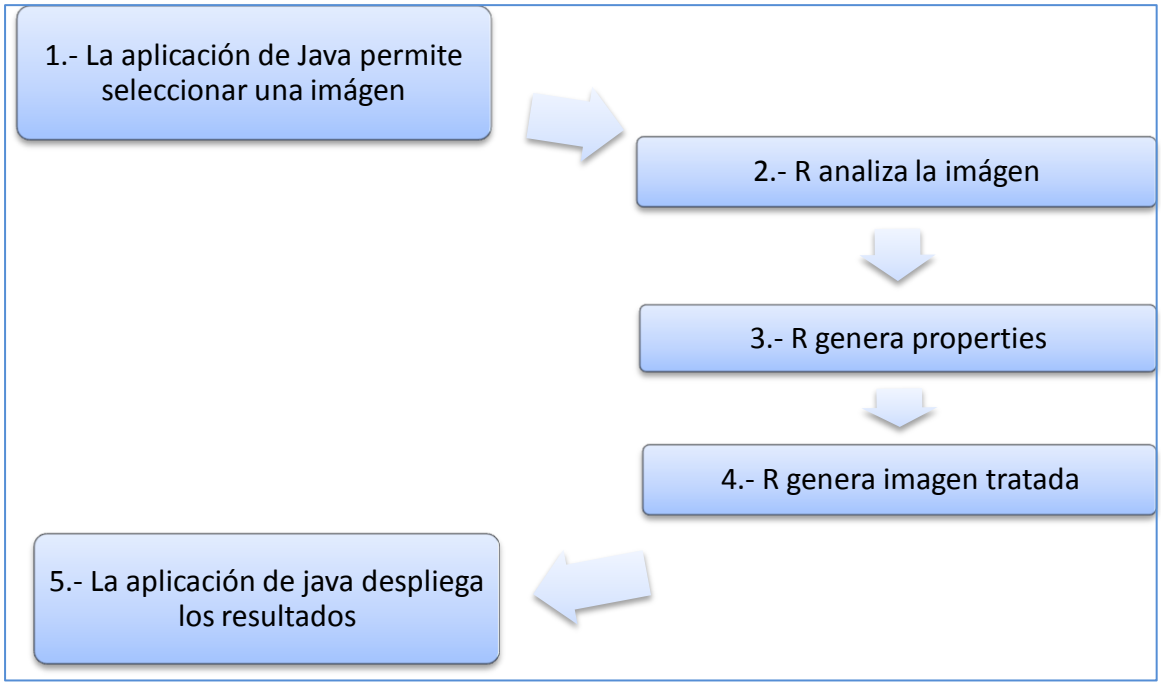

Ilustración 52. Funcionamiento de la aplicación

Internamente R convierte la imagen a blanco y negro. Debido a que las imágenes son obtenidas de un escáner no se consideró eliminar el ruido pues las condiciones de iluminación suelen ser óptimas y el proceso elevaría el tiempo de ejecución de la aplicación.

La imagen es tratada como matriz numérica donde cada pixel es representado con un número que va del 0 al 256 y que representa tonos de obscuros a claros respectivamente **(Ilustración 53).**

Ilustración 53. Imagen en blanco y negro

A partir de la imagen en blanco y negro se detectan los bordes de la hoja **(Ilustración 54)** con el operador de Roberts Cross**.** Este método le aplica dos kernels a la matriz numérica mediante una operación de convolución.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Ejemplo 20. Operador de Roberts Cross.

Al implementarlo, el valor máximo arrojado por la aplicación de dichos patrones es guardado como el valor del borde en ese punto. El punto $E_{x,y}$ en el borde es entonces el máximo de los dos valores derivados de la convolución de los dos kernels en un punto $P_{x,y}$ de la imagen. (Nixon & Aguado, 2008)

$$E_{x,y} = max\{|M^{+} * P_{x,y}|, |M^{-} * P_{x,y}|\} \quad \forall\, x, y \in 1, N-1$$

Ejemplo 21. convolución

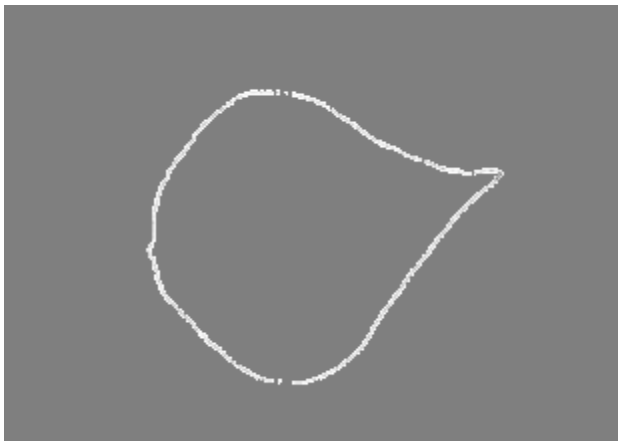

Ilustración 54. Bordes de la hoja

Originalmente el fondo es obscuro y el borde claro pero los valores se invierten para presentarlos al usuario **(Ilustración 55)**.

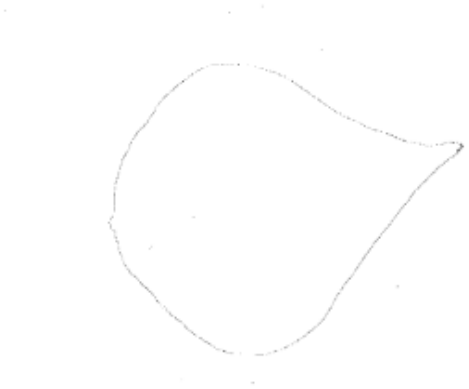

Ilustración 55. Bordes de la hoja con la tonalidad invertida

El ápice es encontrado con un detector de esquinas. Una esquina es el punto en el que la dirección del borde cambia más allá de cierto número de grados. El pixel del borde más alejado a la punta es tomado como la base de la hoja.

Las matrices mostradas a continuación son un ejemplo de secciones que pueden encontrarse en la matriz numérica que representa la imagen, Los valores reales van del 0 al 256 pero se presenta binarizada para su análisis (**Ilustración 56**).

$$\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix} \quad \begin{bmatrix}0&1&0&0\\0&1&0&0\\0&1&0&0\\0&1&0&0\end{bmatrix} \quad \begin{bmatrix}0&1&0&0\\0&0&1&0\\0&1&0&0\\1&0&0&0\end{bmatrix}$$

Zona plana Borde Esquina

Ejemplo 22. Matrices presentes en una matriz binaria.

0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1

0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0

Ilustración 56 borde

Finalmente se realiza un barrido perpendicular al eje formado por la base y la punta para obtener la parte más ancha de la hoja.

Una aplicación java puede ejecutar comandos de R con el API JRI o ejecutar scripts completos con el comando *"R CMD BATCH script.R"*.

En el caso específico de esta aplicación, el script de R recibe como parámetros la ruta a la imagen que el usuario seleccionó, además de sus dimensiones y densidad de pixeles.

Resultados

La aplicación requiere que el usuario tenga a Java y a R apropiadamente instalados. La aplicación consta de un archivo ejecutable que se guarda junto con los scripts de R en una ubicación específica. La aplicación se diseñó para que sea el usuario el que defina la estructura de carpetas y nombres para sus imágenes. La **Ilustración 57** muestra la única ventana de la aplicación.

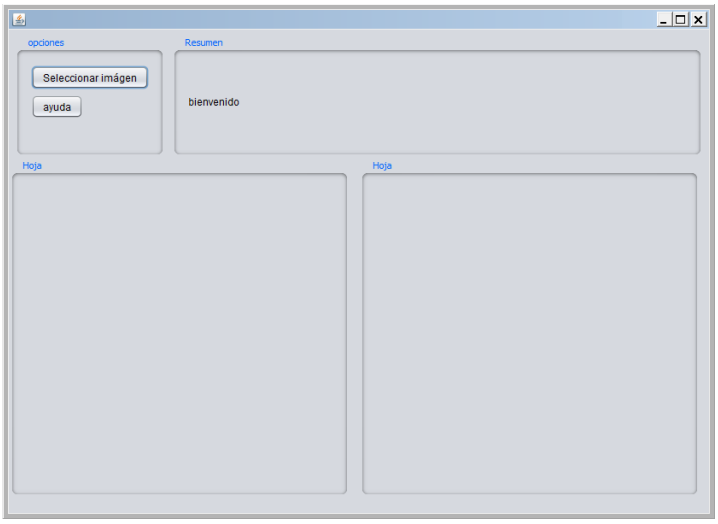

Ilustración 57. Ventana inicial

Para iniciar el análisis de una imagen el usuario selecciona un archivo en formato JPG obtenido desde un escáner presionando el botón "*seleccionar imagen"*. La imagen seleccionada es tratada y el resultado es presentado como texto en la parte superior y gráficamente del lado derecho (**Ilustración 58**).

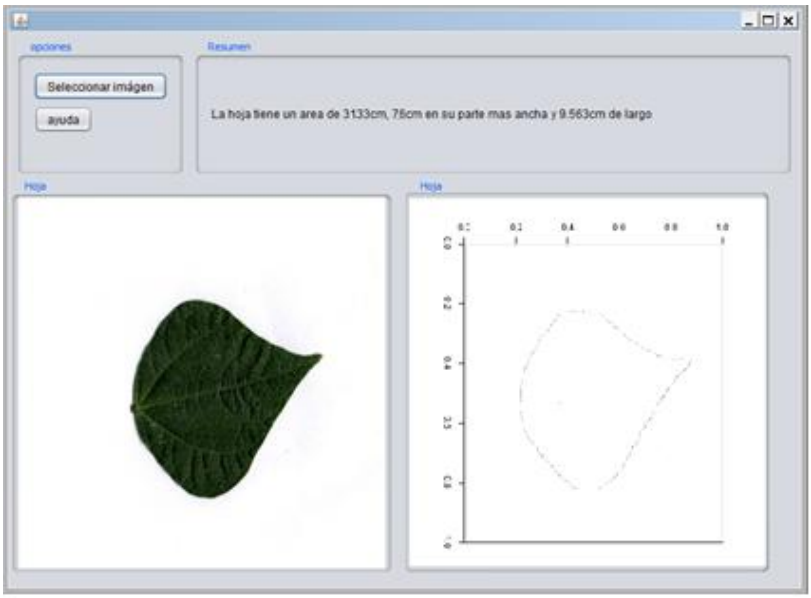

Ilustración 58. Resultados desplegados

Los resultados son almacenados en un archivo de propiedades con el mismo nombre y ubicación que la imagen seleccionada. Si la imagen ya había sido tratada, entonces sólo

despliega los resultados previamente obtenidos; del mismo modo se guarda la imagen tratada digitalmente. Ambos archivos pueden ser revisados por el usuario posteriormente incluso sin utilizar esta herramienta.

Existen opciones gratuitas y comerciales de aplicaciones que permiten realizar un tratamiento digital a las imágenes y obtener sus medidas, pero al ser de propósito general requieren que el usuario realice cada vez configuraciones, pasos y conversiones que se evitan al utilizar una herramienta de uso específico como ésta.

Para el tratamiento de imágenes obtenidas de escáneres no es necesario tomar en cuenta las condiciones de iluminación como en otras técnicas que ocupan cámaras digitales. Tampoco se requiere un fondo de color específico o con patrones.

La aplicación se diseñó para no ser una herramienta intrusiva, es el usuario el que establece su propia estructura de carpetas para almacenar sus archivos digitales.

La aplicación tampoco depende de una marca específica de escáner en tanto que se configure para obtener las imágenes en formato JPG.

La aplicación no está acotada a un tamaño y resolución específicos pero el usuario debe considerar que la velocidad de ejecución de la herramienta depende de las dimensiones y resolución de las imágenes seleccionadas y de las especificaciones del equipo de cómputo; cuanto menor sea la resolución y tamaño de las imágenes, menor será también el tiempo que tome su análisis. El usuario puede configurar su escáner para obtener los archivos con las propiedades que requiera su investigación.

Aplicación móvil para obtener el área foliar de hojas alargadas

La aplicación presentada a continuación conserva los Scripts del lenguaje R desarrollados para la versión de escritorio. La versión móvil permite obtener el largo, ancho y área de hojas alargadas con tan solo tomar una foto, ya que al dispositivo móvil se le conecta un sensor ultrasónico para medir la distancia al objetivo.

El programa realiza el tratamiento digital de las imágenes con el lenguaje R instalado de manera nativa en el celular o tableta que ejecute el Sistema operativo Android.

Introducción

Este trabajo describe el desarrollo de una aplicación para dispositivos móviles que ejecutan el sistema operativo Android que permite obtener el largo, ancho y área de hojas alargadas a las que se les toma una foto contra un fondo de color claro.

Para saber la distancia a la hoja fotografiada se le conecta un sensor ultrasónico a la tableta electrónica o teléfono inteligente que mediante un sistema de sonar obtiene medidas de hasta 6 metros.

Existen aplicaciones móviles que para medir obtienen la distancia al objetivo a partir de la distancia focal, es decir, la separación de las lentes de la cámara durante el zoom automático pero dependen de las capacidades del dispositivo y de las condiciones de iluminación.

La aplicación fue desarrollada originalmente para la hoja de maíz, cuyo tamaño dificulta su medición por otras técnicas, pero puede utilizarse para otros tipos de hojas alargadas.

Materiales, métodos e implementación.

Para el desarrollo del proyecto se le dio prioridad a software gratuito de código abierto y a hardware y materiales de bajo costo**. La ilustración 59** muestra las plataformas utilizadas.

La aplicación utiliza el Lenguaje estadístico R para llevar a cabo el tratamiento de las imágenes digitales. R es la alternativa gratuita y de código abierto a SAS© y Matlab© con la ventaja de ser multiplataforma y de contar con distribuciones específicas para Android e IOS.

La conexión del sensor ultrasónico al dispositivo móvil se realizó con la plataforma para prototipos electrónicos Arduino. Arduino es una interfaz electrónica que permite enviar y recibir señales de control desde una computadora o móvil y hacia dispositivos electrónicos tanto digitales como analógicos.

Ilustración 59. Secciones de la aplicación móvil

La aplicación crea una carpeta en la memoria externa del dispositivo móvil en el que guarda las fotos de las hojas en carpetas para cada día. La captura de la imagen invoca la ejecución de un script del lenguaje R que genera un archivo properties con el ancho, largo y área de la hoja, además de las coordenadas en pixeles que permiten trazar una recta de la base a la punta para su mejor análisis.

El usuario puede por tanto vaciar las fotos posteriormente en un equipo de escritorio y sincronizar los elementos con tan sólo copiar y pegar esta carpeta. La **Ilustración 60** muestra la navegación de la aplicación.

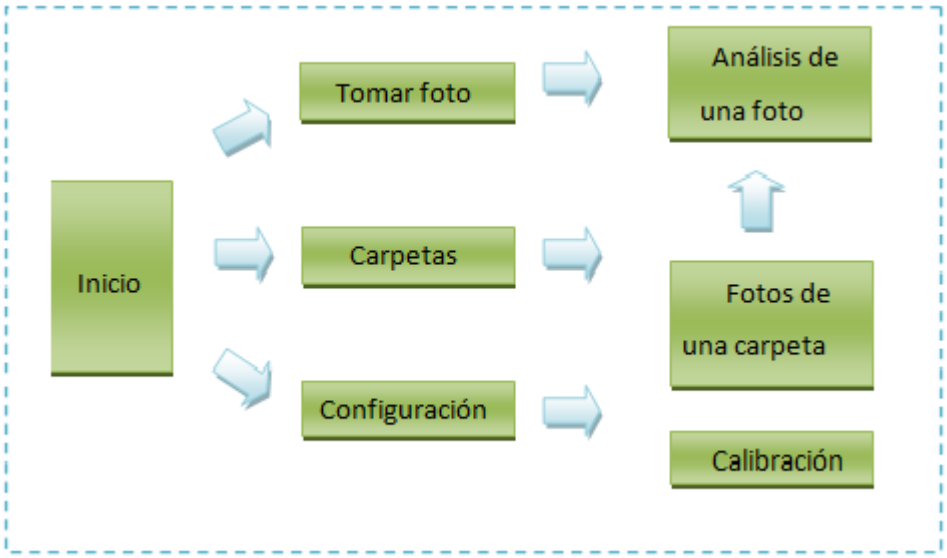

Ilustración 60. Navegación de la aplicación

Los celulares y tabletas tienen cámaras con diferentes capacidades, pero la aplicación cuenta con un botón de calibración el cual, tras tomarle una foto a un circulo de 10 cm de diámetro invoca la ejecución de un Script de R que analiza la foto tomada a la distancia arrojada por el sensor y genera un archivo properties con el número de pixeles que representan un cm. si la foto hubiera sido tomada a un cm. de distancia. De esta manera la aplicación puede usarse indistintamente de la resolución de la cámara que se tenga.

El uso del lenguaje R

El uso de R permite que la lógica de negocios pueda tomarse de desarrollos previos codificados para equipos de escritorio y trasladarla de manera transparente a una plataforma móvil.

La comunicación entre el lenguaje R y java permite que el desarrollo de aplicaciones pueda realizarse por equipos multidisciplinarios, donde algunos integrantes estarán avocados al desarrollo de las estrategias de estadística y de inteligencia artificial y otros especializarse en el desarrollo de las interfaces.

La lógica de negocios codificada con R una sola vez puede reutilizarse para varias aplicaciones no necesariamente desarrolladas con Java, tanto móviles como de escritorio o web.

La instalación de R en sistema operativo Android requiere privilegios de superusuario. La instalación del lenguaje R y la obtención de permisos de superusuario deben realizarse por un profesional.

La instancia del lenguaje R instalada puede compartirse para múltiples aplicaciones

Para que R pueda interpretar la imagen en formato bitmap como matriz numérica necesita los paquetes bmp y pixmap. Para dispositivos móviles no puede instalarse paquetes extra pero puede descargarse el código fuente de los paquetes e importarlo al script propio con el comando source.

```
source(' read-bmp.R');
```

Ejemplo 23. Comando R para importar script

El paquete bmp convierte la imagen a blanco y negro y la retorna como matriz numérica. El siguiente paso es el cálculo del valor promedio de tonos obscuros para la segmentación y obtención del área. Detecta las dos esquinas correspondientes a la base y a la punta de la hoja y con la ecuación de la recta calcula dicha distancia que corresponde al largo. Sobre este eje se hace un barrido perpendicular para detectar la zona más ancha.

Los valores obtenidos en términos de pixeles son convertidos a centímetros y el resultado es almacenado en un archivo properties para que la aplicación de Android despliegue su contenido como resultado.

La **Ilustración 61** muestra el diagrama de clases del módulo de procesamiento estadístico. La capa de servicios es la lista de funciones disponibles en este módulo y al ser el único punto de acceso es bajo el acoplamiento con la interface gráfica.

El módulo requiere clases Helper para crear carpetas, imágenes y archivos properties. El helperR es el que invoca la ejecución de los scripts. El helper Android permite acceder a las ubicaciones relativas del dispositivo móvil, como por ejemplo la memoria SD, ya que la ubicación de las carpetas del SO cambia para cada modelo de celular.

Ilustración 61 diagrama de clases del módulo estadístico

La conexión del sensor ultrasónico

El sensor ultrasónico (**Ilustración 62**) permite obtener la distancia a un objetivo mediante un sistema de sonar en un angulo de 40 grados en un rango de hasta 6 metros. Opera con 5 volts.

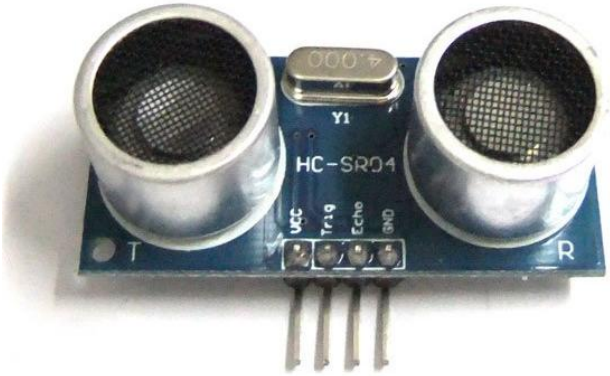

Ilustración 62. Sensor ultrasónico

Cada marca y modelo de sensor podría tener un protocolo diferente y para su uso podría requerirse una biblioteca. También las diferentes ediciones del entorno de desarrollo usan distintas bibliotecas.

Durante la fase de desarrollo del proyecto y solamente para equipos de escritorio se necesita agregar la biblioteca RxTx.dll al classpath de la aplicación. Cuando la estrategia de comunicación con el sensor se encuentre diseñada, será momento de trasladarla a un entorno móvil (**Ilustración 63**).

Ilustración 63. Conexión del sensor ultrasónico a un equipo de escritorio

Las dos entradas de comunicación del sensor se conectan a dos pines de la tableta Arduino, que a su vez se conecta al dispositivo móvil (**Ilustración 64**).

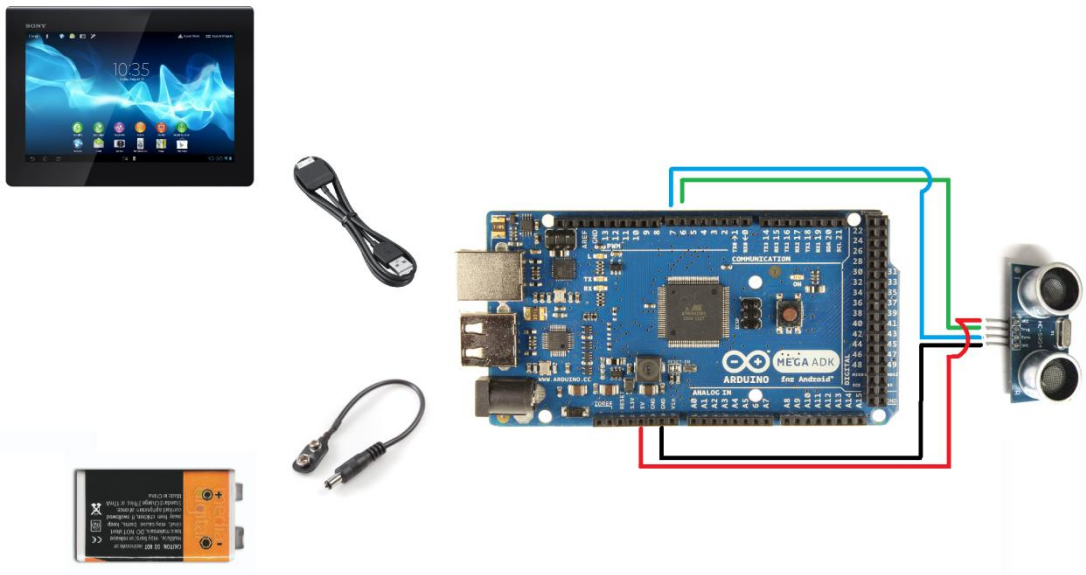

Ilustración 64. Conexión del sensor ultrasónico a un dispositivo móvil

Los programas de arduino poseen un método loop que se ejecuta en un bucle mientras la tableta se encuentre encendida. En este método la distancia marcada por el sensor es enviada a la computadora o dispositivo móvil para su uso.

La tableta con el programa ya cargado se conecta al puerto de datos del celular o tableta, dependiendo del modelo de dispositivo. Cuando este en producción deberán suministrársele 5 volts.

Resultados

La aplicación almacena las fotos tomadas en carpetas para cada día y los nombres del archivo se obtienen a partir de la hora de la captura. Esto se diseñó para que la sincronización con fotos en un equipo de equipo de escritorio fuera más sencilla pues basta con mezclar los contenidos de las carpetas y archivos que tendrán nombres únicos.

La aplicación cuenta, en la sección de configuración, con un botón que crea la estructura inicial de carpetas y que carga a la memoria externa del dispositivo móvil la última versión de los scripts de R. La **Ilustración 65** muestra la ejecución de la aplicación.

Ilustración 65. Ventana inicial de la aplicación

La sección de toma de fotografía despliega en primer plano la imagen a capturar. Del lado derecho de la pantalla se muestra la inclinación de la tableta que es tomada de su acelerómetro. Se sugiere que las fotos sean tomadas con la tableta totalmente horizontal pero no es obligatorio para su uso en campo y en caso de evitar que la sombra del dispositivo se proyecte sobre la muestra.

Debajo una barra azul crece y disminuye marcando la distancia al objetivo. Al final aparece el botón con el que se toma la fotografía (**Ilustración 66**).

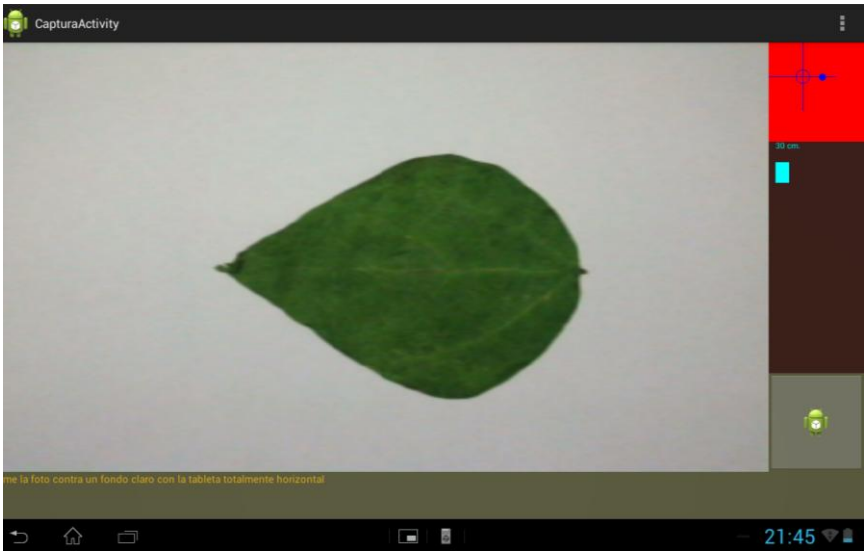

Ilustración 66. Captura de las hojas

Tras la toma de la imagen se despliega la información de ancho, largo y área de la hoja y una vista de la foto sobre la que se dibujan de color azul un eje vertical y uno horizontal en los lugares más anchos y largos de la hoja (**Ilustración 67**). Toda esta información, incluyendo las coordenadas de los ejes, es tomada del archivo de propiedades creado por el lenguaje R.

Ilustración 67. Captura del resultado

Para sostener la tableta Arduino se realizó una estructura de acrílico. Bajo las condiciones adecuadas puede fotografiarse a las hojas de una planta sin necesidad de cortarlas. Basta con pasar la hoja de la planta a través de una ranura en el centro de una hoja blanca. También puede presionarse la muestra con acetato transparente para mejorar la calidad de las fotos (**Ilustración 68**).

Ilustración 68. Uso en campo

El script de tratamiento de la hoja se mantuvo lo más simple posible para que su ejecución fuera lo más veloz posible. Una imagen de 400 por 600 pixeles demora aproximadamente 4 segundos en analizarse incluyendo los tiempos de lectura y escritura de archivos.

La **Ilustración 69** muestra el ejercicio realizado sobre un cuadrado de 10 centímetros por lado que arroja como resultado que mide 100.6412 cm cuadrados.

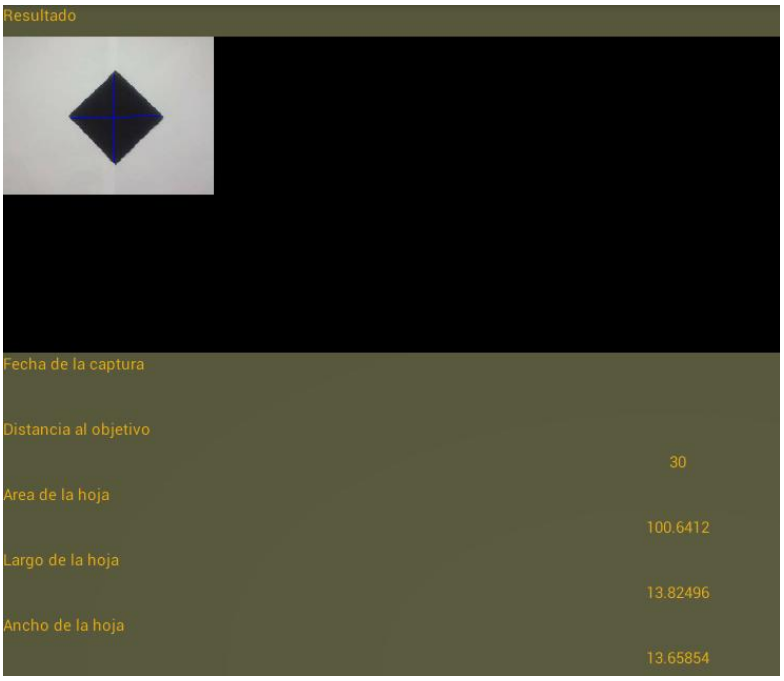

Ilustración 69. Ejercicio con un cuadrado de 10cm por 10 cm

Para un ejercicio realizado con hojas de frijol se midió primero contra una hoja milimétrica su área resultando en 37.19 cm cuadrados. La **Ilustración 70** muestra la captura realizada por el dispositivo y que arroja 36.61 cm cuadrados como resultado, siendo la diferencia del 1.6%.

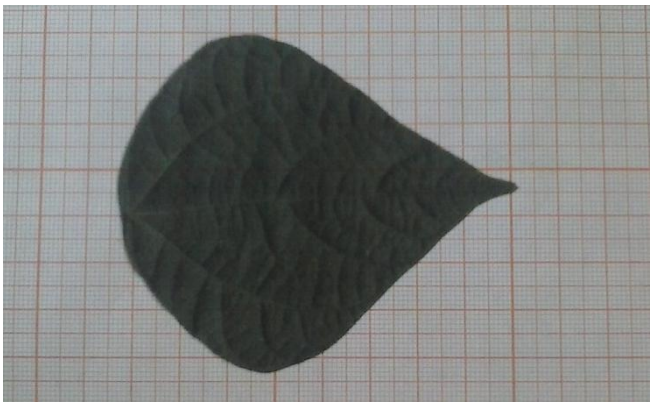

Ilustración 70. Estimación manual

Discusión

El sistema operativo Android, la plataforma Arduino y el lenguaje R son gratuitos y de código abierto. Los dispositivos móviles que ejecutan Android se encuentran entre los de menor costo y la tableta Arduino puede construirse por el usuario o comprarse a muy bajo precio. Igualmente los entornos de desarrollo de Android, R y Arduino son gratuitos.

El uso de la tableta Arduino requiere conocimientos básicos del lenguaje C del que existe gran documentación. Arduino se beneficia de las cualidades de modularidad, seguridad y robustez del lenguaje C.

Para que un sensor pueda ser utilizado con Arduino su fabricante debe proveer la biblioteca de vínculos dinámicos necesaria o proporcionar información de su protocolo de comunicación. Hay una gran variedad de sensores analógicos y digitales listos para ser utilizados con Arduino.

La comunicación de la tableta Arduino y la aplicación nativa se realiza mediante paquetes de datos de tamaño variable que se envían fragmentados. La aplicación nativa que recibe los fragmentos debe unirlos en estructuras bien definidas.

También hay comunicación del celular hacia la tableta Arduino lo que suma las capacidades de automatización de la interface electrónica a la potencia de cálculo del dispositivo Android.

Debe considerarse que la tableta Arduino es compatible con equipos cuyo sistema operativo es versión 3 o superior y que en producción será necesario suministrarle 5 volts.

La instalación del lenguaje R ocupa 170 megabytes de la memoria interna del dispositivo móvil, si no se cuenta con espacio suficiente, puede instalarse en la memoria externa.

Ante la imposibilidad de instalar paquetes extra al lenguaje R que se ejecuta en Android, pueden descargarse como código fuente e incorporarse como parte del script a ejecutar.

Para el uso de la interface R CMD BATCH puede implementarse como estrategia el uso de una carpeta de archivos temporales en los que la aplicación de Android almacene los datos a analizar para que sean accesibles a R que también almacenará el resultado en una localidad accesible a la aplicación nativa. Otra posibilidad es que el script a ejecutarse se cree dinámicamente por la aplicación nativa cada vez. El primer esquema no representa inconvenientes en el caso de aplicaciones que requieren el guardado de información histórica.

Para el desarrollo de la plataforma se requieren privilegios de superusuario. En los sistemas linux el usuario root es el que posee todos los privilegios de ejecución, lectura y escritura de archivos. Este usuario no se encuentra disponible por default en dispositivos móviles pero puede obtenerse mediante aplicaciones que modifican el sistema operativo. Dicho proceso es delegado al dueño del equipo pues el proceso es diferente para cada modelo de dispositivo.

Una segunda opción para instalar R es mediante la instalación previa de la aplicación GNU root que le permite al sistema operativo Android instalar programas mediante comandos como en cualquier Linux.

La conexión de los sensores directamente al dispositivo móvil y la instalación de la versión para Android del lenguaje R favorecen la portabilidad y usabilidad de la plataforma.

Debe tenerse en cuenta las condiciones de humedad y temperatura en las que será utilizado un dispositivo electrónico al momento de ser diseñado.

Conclusiones

La instalación del lenguaje R en un celular o tableta permite trasladar la lógica de negocios validada de una aplicación de escritorio a un dispositivo móvil. Su instalación requiere una serie de pasos técnicos realizados por personal especializado que influyen en la arquitectura de la aplicación y en las vías de distribución del producto ya terminado.

Aunque el rendimiento de un dispositivo móvil está limitado por su hardware, arquitectónicamente su funcionamiento es similar al de un equipo de escritorio. De igual manera la plataforma para dispositivos móviles presentada en este trabajo no difiere estructuralmente de lo que es posible realizar en un equipo de escritorio; las diferencias son técnicas.

Tanto R como Java y C son lenguajes ampliamente utilizados en el ámbito académico y de investigación. R, Android y Arduino son plataformas abiertas y gratuitas que posibilitan el desarrollo de prototipos electrónicos a bajo costo.

El campo de las aplicaciones móviles de interés científico es naciente. El uso de dispositivos móviles se encuentra en crecimiento al igual que su capacidad de procesamiento. La integración del lenguaje R y la tableta Arduino en el sistema operativo Android crea nuevas herramientas y nuevos campos de aplicación para los dispositivos móviles.

ANEXO A

El desarrollo de aplicaciones para el S.O. Android.

En este anexo se explican los pasos para configurar y desarrollar una aplicación móvil. La estructura de carpetas y los archivos que incluye una aplicación móvil son referidos a lo largo de la tesis.

El asistente para la creación de aplicaciones inicia con la especificación del nombre y ubicación del proyecto **(ilustración 71**). Además hay que establecer:

- Application Name: Nombre de la aplicación que aparecerá al ejecutarse, en los iconos y en el play store.
- Project Name: Nombre del proyecto en eclipse.
- Package Name: nombre del paquete en el que se guardarán las clases.
- Minimum Required SDK: versión mínima de Android que soportará la aplicación.
- Target SDK: versión máxima de Android que soportará la aplicación.
- Compile with: versión del SDK con la que se compilará la aplicación.

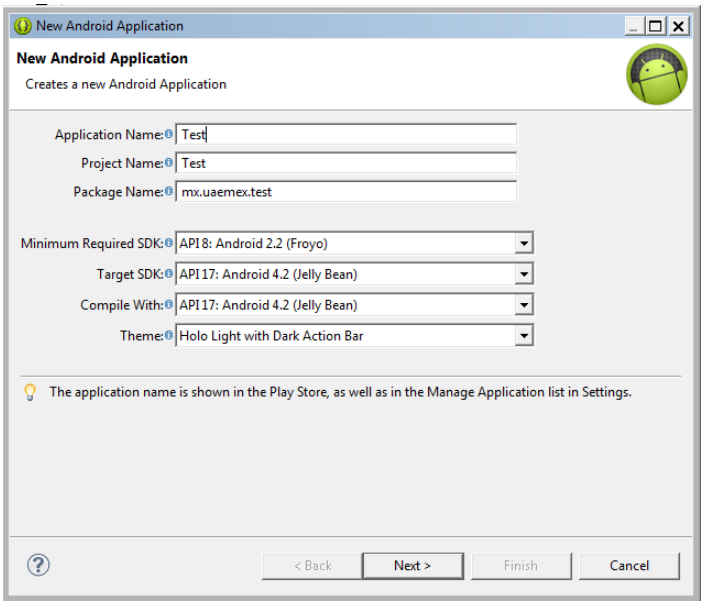

Ilustración 71. Nueva aplicación Android

Al presionar siguiente aparece una ventana de configuración con la opción “mark this project as library” que se deja inactiva cuando se desarrollan interfaces gráficas y activada cuando se desarrollan módulos reutilizables o bibliotecas (**Ilustración 72**).

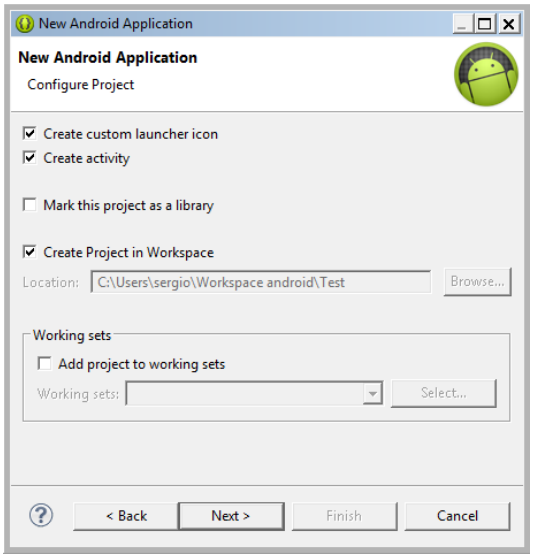

Ilustración 72. Nueva aplicación Android, crear Activity por default

El siguiente paso es cargar un icono para la aplicación, de otro modo se utiliza el que el editor incluye por default (**Ilustración 73**).

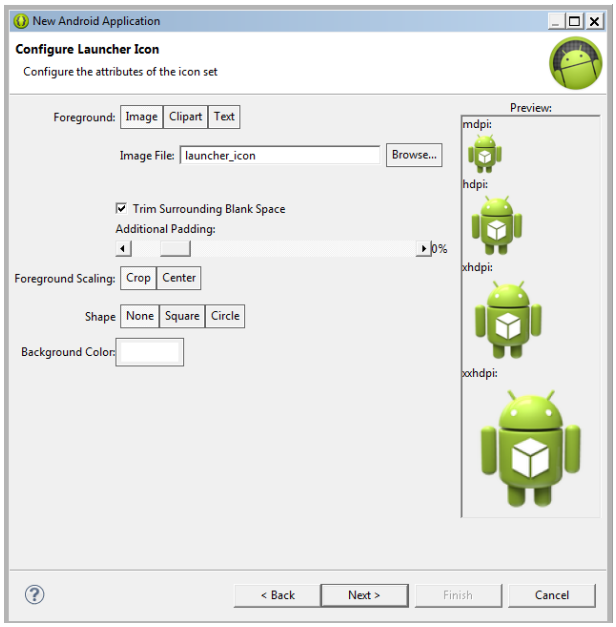

Ilustración 73. Nueva Aplicación Android, seleccionar icono

En el siguiente paso puede seleccionarse la opción “crear activity” que genera una ventana en blanco (**Ilustración 74**).

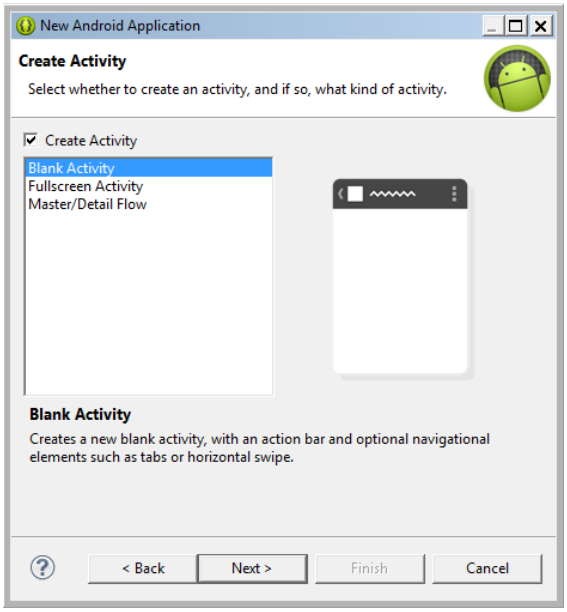

Ilustración 74. Nueva aplicación Android, tipo de Activity

De seleccionarse esa opción hay que especificar el nombre de la ventana (**Ilustración 75**).

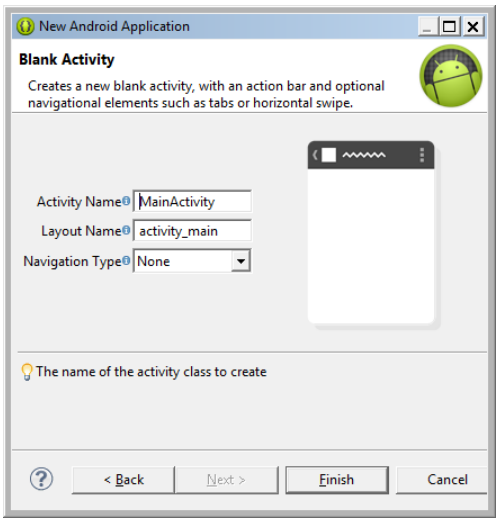

Ilustración 75. Nueva Aplicación Android, Nombre de la Activity

Finalmente presionando el botón finish se crea el proyecto. La **Ilustración 76** muestra la estructura de carpetas de un proyecto para el SO. Android:

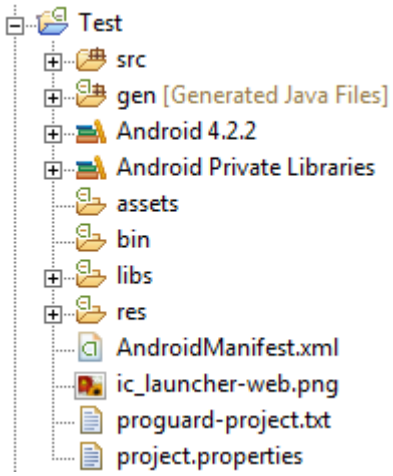

Ilustración 76. Estructura de carpetas de la aplicación

- Src: clases de Java
- Gen: código generado al compilar el proyecto
- Assets: Recursos tales como fotos, música, scripts, de configuración, etc. accesibles para la aplicación.
- Bin: ejecutable generado tras la compilación.
- Libs: bibliotecas
- Res: vistas

- AndroidManifest.xml: archivo descriptor del proyecto.

Activities.

Las ventanas de una aplicación Android se llaman Activities. Al utilizar el editor de creación de Activities se genera un archivo XML que describe la interface en la carpeta res/layout. También se crea un archivo XML en la carpeta res/menu que genera la barra de menús de cada ventana. Por último, se genera una clase Java en la carpeta SRC que recibirá las acciones de los botones de la ventana.

La aplicación creada con las instrucciones del anexo A se genera con una ventana de ejemplo. El código XML que la modela está en res/layout/activity_main.xml.

Esta vista solo contiene un elemento de tipo TextView que muestra el texto *"HelloWorld!"* (Ejemplo 1).

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity" >
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/hello_world" />
</RelativeLayout>
```

Ejemplo 1. ActivityMain.xml

La **Ilustración 77** muestra la ejecución del código.

Ilustración 77. Hello world Android

La propiedad text del elemento TextView establece la cadena de texto a mostrar. Ésta está escrita en res/values/Strings.xml (Ejemplo 2) bajo el identificador “hello_world”. Siempre que se use el prefijo @string se refiere a un elemento de este archivo XML.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">Test</string>
<string name="action_settings">Settings</string>
<string name="hello_world">Hello world!</string>
</resources>
```

Ejemplo 2. Strings.xml

El componente RelativeLayout tiene varias propiedades que apuntan a valores con el prefijo @dimen. Estos valores se establecen en el archivo res/values/dimens.xml (Ejemplo 3).

```
<resources>
<!-- Default screen margins, per the Android Design guidelines. -->
<dimen name="activity_horizontal_margin">16dp</dimen>
<dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

Ejemplo 3. Dimens.xml

Un botón puede agregarse a la interface con tan sólo arrastrarlo desde la paleta al lugar deseado de la interface, habrá que editar su rotulo al texto deseado (**Ilustración 78**).

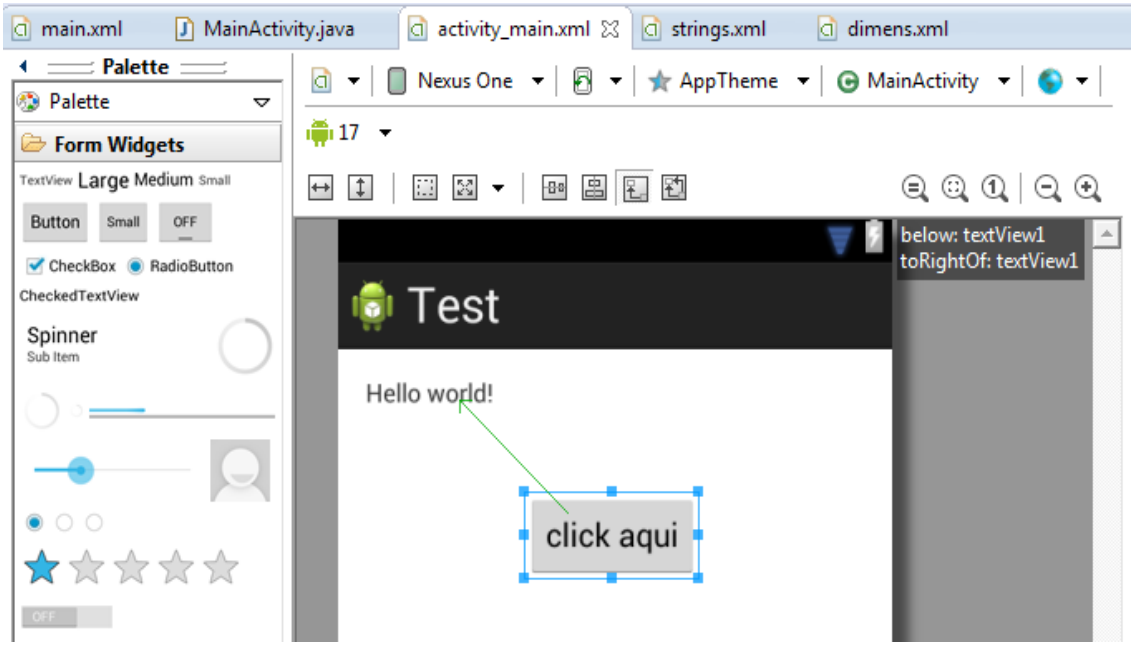

Ilustración 78. Agregar un botón a una Activity

Por cada Activity se crea una clase de Java en la carpeta src que recibe las solicitudes de las interfaces. El paquete en el que se almacenan estas clases se establece durante la creación del proyecto (véase Anexo A). En este caso, para la Activity ActivityMain.xml se genera la clase mx.uaemex.test.MainActivity en la que puede agregarse un método a ejecutarse cuando se presione el botón. El ejemplo 4 establece un método *onClickMethod* que cambia el texto del componente textView para desplegar la hora.

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
    }
    @Override
    publicbooleanonCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
    }
    public void onClickMethod(View v){
            TextViewtextView = (TextView)findViewById(R.id.textView1);
            textView.setText("fecha: "+new Date());
    }
}
```

Ejemplo 4. MainActivity.java

En el ejemplo 4, el método findViewById es utilizado para acceder a una instancia del objeto *textView1*. La clase gen.<package>.<Project>.R guarda un apuntador a los elementos gráficos de la aplicación; Esta clase es autogenerada por eclipse cada vez que se compila el proyecto. Así que cada que se modifique alguna interfaz gráfica hay que compilar el proyecto antes de poder hacer uso de la clase R.

El ejemplo 5 muestra la línea de código necesaria para establecer el método que un botón ejecutará al accionarlo.

```
android:onClick="onClickMethod"
```

Ejemplo 5. Establecer la acción de un botón.

La aplicación esta lista para probarse. Para ello hay que conectar el dispositivo Android al puerto USB de la computadora y presionar el botón “run” de la barra de herramientas. También puede probarse en una maquina virtual (**Ilustración 79**).

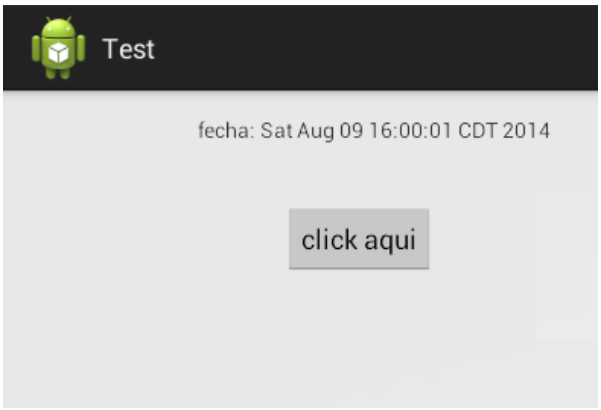

Ilustración 79. Ejecución del ejemplo

El archivo AndroidManifest.xml.

El archivo AndroidManifest.xml es el archivo descriptor de la aplicación. Todas las ***activities*** del programa son dadas de alta en él.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="mx.uaemex.test"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdandroid:minSdkVersion="8" android:targetSdkVersion="17" />
<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
    android:name="mx.uaemex.test.MainActivity"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
</application>
</manifest>
```

Ejemplo 6. AndroidManifest.xml

La etiqueta uses-sdk establece la versión mínima y máxima de SDK que la aplicación soporta y que fue especificado durante la creación del proyecto (véase anexo A).

La etiqueta "***activity***" se agrega por cada ***Activity*** que se defina para la aplicación. Sólo una de las etiquetas activity tendrá anidada la etiqueta ***"Android.intent.action.MAIN"*** que especifica la activity principal, es decir, la primera ventana que será desplegada al ejecutarse el programa.

Este archivo también especifica la orientación de las ventanas, la propiedad ***Android:screenOrientation*** puede tomar los valores ***potrait*** o ***landscape*** para vertical u horizontal respectivamente.

En este archivo se establecen los permisos requeridos por la aplicación, tales como usar la cámara o guardar archivos en la memoria externa. La etiqueta uses-permission se agregan inmediatamente después de la etiqueta uses-sdk.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Ejemplo 7. Permisos para el uso de la cámara y memoria SD

Otra herramienta de las ADT es "***emulator.exe***" que permite la creación de máquinas virtuales del S.O. Android para probar las aplicaciones.

Anexo B: Código fuente

Archivo AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="mx.uaemex.greenscanner"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="14"
      android:targetSdkVersion="18" />

   <uses-permission android:name="android.permission.CAMERA" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

   <uses-feature android:name="android.hardware.camera" />
   <uses-feature android:name="android.hardware.camera.autofocus" />
   <uses-feature android:name="android.hardware.usb.accessory" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <uses-library android:name="com.android.future.usb.accessory" >
      </uses-library>

      <activity
         android:name="mx.uaemex.greenscanner.MainActivity"
         android:label="@string/app_name"
         android:screenOrientation="landscape" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
         <intent-filter>
  <action android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
         </intent-filter>

         <meta-data
         android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
         android:resource="@xml/accessory_filter" />
```

```
    </activity>
    <activity
      android:name="mx.uaemex.greenscanner.CapturaActivity"
      android:label="@string/title_activity_captura"
      android:screenOrientation="landscape" >
    </activity>
    <activity
      android:name="mx.uaemex.greenscanner.CapturaAnalisisActivity"
      android:label="@string/title_activity_captura_analisis"
      android:screenOrientation="landscape" >
    </activity>
    <activity
      android:name="mx.uaemex.greenscanner.ExplorarCategoriaActivity"
      android:label="@string/title_activity_explorar_categoria"
      android:screenOrientation="landscape" >
    </activity>
    <activity
      android:name="mx.uaemex.greenscanner.ExplorarListadoActivity"
      android:label="@string/title_activity_explorar_listado"
      android:screenOrientation="landscape" >
    </activity>
    <activity
      android:name="mx.uaemex.greenscanner.ExplorarFotoActivity"
      android:label="@string/title_activity_explorar_foto"
      android:screenOrientation="landscape" >
    </activity>
    <activity
      android:name="mx.uaemex.greenscanner.CalibrarActivity"
      android:label="@string/title_activity_calibrar"
      android:screenOrientation="landscape">
    </activity>
  </application>
</manifest>
```

Funcion *creaProperties* del lenguaje R que crea un archivo properties

```
creaProperties = function(rutaProperties, keySet, valueSet){
  sink (rutaProperties)
  size=length(keySet)
  for(i in 1:size){
    cat("\n")
    cat(keySet[i])
    cat(" = ")
    cat(valueSet[i])
  }
  sink()
}

keySet = c('resultado1', 'resultado2' )
valueSet = c(5,7)

creaProperties(rutaProperties,keySet, valueSet)
```

Método *copyFromAssetsToSd* que copia un archivo de la carpeta assets a la memoria externa.

```
public boolean copyFromAssetsToSD(String folder, String file){
    InputStream inStream = null;
    OutputStream outStream = null;
     int length =0;
    AssetManager assetManager = getAssets();
    String from = file;
    if(folder!=null){
            if(!folder.isEmpty()){
                    from = folder+"/"+from;
            }
    }
    try{
            inStream = assetManager.open(from);
            String outPath = getApplicationPath(folder,file);
            outStream = new FileOutputStream(outPath);
            byte[] buffer = new byte[1024];
            while ((length = inStream.read(buffer)) > 0){
                    outStream.write(buffer, 0, length);
                    }
            inStream.close();
            outStream.close();
            Log.e("Configuracion activity", " OK");
            return true;
    }catch(IOException e){
             Log.e("Configuracion activity", e.getMessage());
    }
    return false;
}
```

Método validarArchivoExista de la clase FileHelper que recibe como parámetro la ruta a un archivo del que se requiere validar su creación. El método checa y espera un tiempo especificado por la variable jump. La variable stop establece el tiempo máximo que el método intentara validar si el archivo existe.

```
public static void validarArchivoExista(String path, int stop, int jump){
    int start =0;
    try {
            File file = new File(path);
            Log.e("FileHelper", "esperar, archivo existe? " + path+file.exists());
            while(!file.exists()&&start<stop){
                    start +=jump;
                    Thread.sleep(jump);
                    Log.e("FileHelper", "esperar, archivo existe? " + path+file.exists());
            }
    } catch (InterruptedException e) {
            Log.e("main pausa", "error esperando la creacion de "+path);
    }
}
```

Método OpenAccessory

```
UsbManager mUsbManager = UsbManager.getInstance(this);
Private void open Accessory(UsbAccessory accessory) {
        mFileDescriptor = mUsbManager.openAccessory(accessory);
        if (mFileDescriptor != null) {
                mAccessory = accessory;
                FileDescriptorfd = mFileDescriptor.getFileDescriptor();
                mInputStream = newFileInputStream(fd);
                mOutputStream = newFileOutputStream(fd);
                Thread thread = newThread(null, this, "OAO");
                thread.start();
                Log.d(“OAO”, "accessory opened");
        } else {
                Log.d(“OAO”, "accessory open fail");
        }
  }
```

Método closeAccesory

```
Private void closeAccessory() {
        try {
                if (mFileDescriptor != null) {
                        mFileDescriptor.close();
                }
        } catch (IOException e) {
        } finally {
                mFileDescriptor = null;
                mAccessory = null;
        }
    }
```

Metodo que descomprime un archivo *.tar*

```
public static boolean unTar(String folder,String fileName){
    String file_path = getApplicationPath(folder,fileName);
    validarArchivoExista(file_path,40000,1000);
    try {
            Rhelper.proccesCommand("tar xf  "+file_path+" -C /data/local/gcc");
            return true;
    } catch (Exception e) {
            Log.e("FileHelper",  e.getLocalizedMessage());
    }
    return false;
}
```

Bibliografía

Android Open Source Project. (2011). *Android Open Accessory Protocol*. Retrieved junio 1, 2013, from https://source.android.com/accessories/protocol.html

Android Open Source Project. (2012). *Accessory Development Kit 2012 guide*. Retrieved junio 1, 2013, from http://developer.android.com/tools/adk/index.html

Android Open Source Project. (n.d.). *Android Debug Bridge*. Retrieved junio 1, 2013, from http://developer.android.com/tools/help/adb.html

Android Open Source Project. (n.d.). *Android development Tools*. Retrieved junio 1, 2013, from http://developer.android.com/tools/sdk/eclipse-adt.html

Android Open Source Project. (n.d.). *Introducing Android*. Retrieved junio 1, 2013, from http://www.android.com/intl/es/about

Apple inc. (n.d.). *What is IOS*. Retrieved junio 1, 2013, from http://www.apple.com/ios/what-is

Arduino. (n.d.). Retrieved 09 14, 2014, from htto://arduino.cc

Atmel. (2012). *8-bit Atmel microcontroller with 64k/128k/256k Bytes In-system Programmable Flash.* Retrieved from http://www.atmel.com/Images/doc2549.pdf

Atmel corporation. (1999). In *AVR RISC MICROCONTROLLER* (pp. 13-9). San Jose California.

Bayle, J. (2013). C Programming for Arduino. Packt Publishing.

booch, g. (1996). *Análisis y diseño orientado a objetos con aplicaciones.* México: Addison Wesley Longman.

Doebelin, E. E. (2005). *Sistemas de medición e instrumentación.* México: Mc Graw Hill.

Evans, m. e. (2013). *Arduino in action.* Manning.

Flynn, I., & McHoes, A. (n.d.). Sistema Operativo Unix. In *Sistemas Operativos.* International Thomson Editores.

Forestry Suppliers. (n.d.). *Model CI-202L Portable Laser Leaf Area Meter*. Retrieved noviembre 14, 2014, from http://www.forestry-suppliers.com/index1.asp

Google. (n.d.). *About google*. Retrieved junio 1, 2013, from https://www.google.com.mx/intl/es/about

Gooogle Inc. (2013, noviembre 27). *Android 4.4 Compatibility Definition.* Retrieved from http://static.googleusercontent.com/media/source.android.com/es//compatibility/android-cdd.pdf

grosjean, p. (2013). *r wiki*. Retrieved septiembre 10, 2013, from http://rwiki.sciviews.org/doku.php?id=getting-started%3ainstallation%3aandroid

Hornik, K. (n.d.). *Frequently Asked Questions on R*. Retrieved junio 1, 2013, from http://CRAN.R-project.org/doc/FAQ/R-FAQ.html

IDC. (2013). *Wolrdwide Smartphone OS Market Share*. Retrieved from http://www.idc.com/prodserv/smartphone-os-market-share.jsp

John Deere. (2014, febrero 12). *John Deere Introduces SeedStar Mobile for improving Planter Performace*. Retrieved from http://www.deere.com/en_US/corporate/our_company/news_and_media/press_releases/2014/agriculture/2014feb12_seedstar_planter.page

Kabacoff, R. (2011). R in Action, data analisys and graphics whit R. Manning.

Komatineni, S. (2012). *Pro Android 4.* Apress.

Linux org. (n.d.). *Information and resources about the Linux Operating System*. Retrieved junio 1, 2013, from http://www.linux.org

Lugo-espinosa, O. (2013). Ibfieldbook, an integrated breeding field book for plant breeding. *Fitotecnia Méxicana Vol 36*, 201-108.

mathWorks. (n.d.). *MatLab*. Retrieved junio 1, 2013, from http://www.mathworks.es/products/matlab

Maxim integrated. (2014). *MAX3421E*. Retrieved from USB Peripheral/host Controller whith SPI interface: http://www.maximintegrated.com/en/products/interface/controllers-expanders/MAX3421E.html/tb_tab0

Mednieks, Z., & al, e. (2011). Programming Android. O'Reilly.

Microsoft. (n.d.). *new features of Windows Phone*. Retrieved junio 1, 2014, from http://www.windowsphone.com/en-us

Milward Brown. (2013, nov 25). *Estudio de Usos y hábitos de Dispositivos móviles en México.* Retrieved from www.millwardbrown.com

Nixon, M., & Aguado, A. (2008). Feature Extraction and image processing. Elsevier.

Oracle. (2013, diciembre 3). *R technologies from Oracle*. Retrieved from Bringing the power of R to the Enterprise: http://www.oracle.com/technetwork/database/database-technologies/r/r-technologies/r-offerings-1566363.html

Primefaces. (n.d.). *Why Primefaces*. Retrieved junio 1, 2014, from http://primefaces.org/whyprimefaces

R Core Team. (n.d.). *An Introduction to R, Notes on R: A Programming Environment for Data Analysis and Graphics.* Retrieved junio 1, 2013, from http://cran.r-project.org/doc/manuals/R-intro.pdf

Ramírez Vique, R. (n.d.). *Métodos para el dasarrollo de aplicaciones móviles.* FUOC. Fundactó per a la Universitat Oberta de Catalunya.

RStudio. (n.d.). *About RStudio*. Retrieved junio 1, 2014, from http://www.rstudio.com/about

SAS. (n.d.). *About SAS*. Retrieved junio 1, 2013, from http://www.sas.com/en_us/company-information.html

Sheng, L. (1999). The Java Native Interface Programmer's Guide and Specification. Addison-Wesley.

The Eclipse Foundation. (2014). *The eclipse foundation open source community website*. Retrieved junio 1, 2014, from http://www.eclipse.org

The R project. (n.d.). *The R project for statistical computing*. Retrieved junio 1, 2013, from http://www.r-project.org

TIOBE. (2014). *TIOBE programming comunity index*. Retrieved from http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Valvano, J. W. (2004). *Introduction to embedded microcomputer systems. Motorola 6811 and 6812 simulation.* méxico: Thomson.

Vera, P. (2013). Metodolgía de Modelado de Aplicaciones Web Móviles Basada en Componentes de Interfaz de Usuario. *Argentine Symposium on Software Engineering*, 288.

W3C. (2010, diciembre 14). *Mobile Web Application Best Practices.* Retrieved from http://www.w3.org/TR/mwabp/

Wikipedia. (2013, agosto 30). *Serial Peripheral Interface*. Retrieved from http://es.wikipedia.org/wiki/Serial_Peripheral_Interface

wikipedia. (n.d.). *Arduino*. Retrieved junio 1, 2014, from https://es.wikipedia.org/wiki/Arduino

Wolfson, M. (2013). Android Developer Tools Essentials. O'Reilly.

Zechner, M., & Green, R. (n.d.). *Beginning Android Games.* Apress.