Programación Móvil

Unidad 2: Arquitecturas y Entornos de Desarrollo

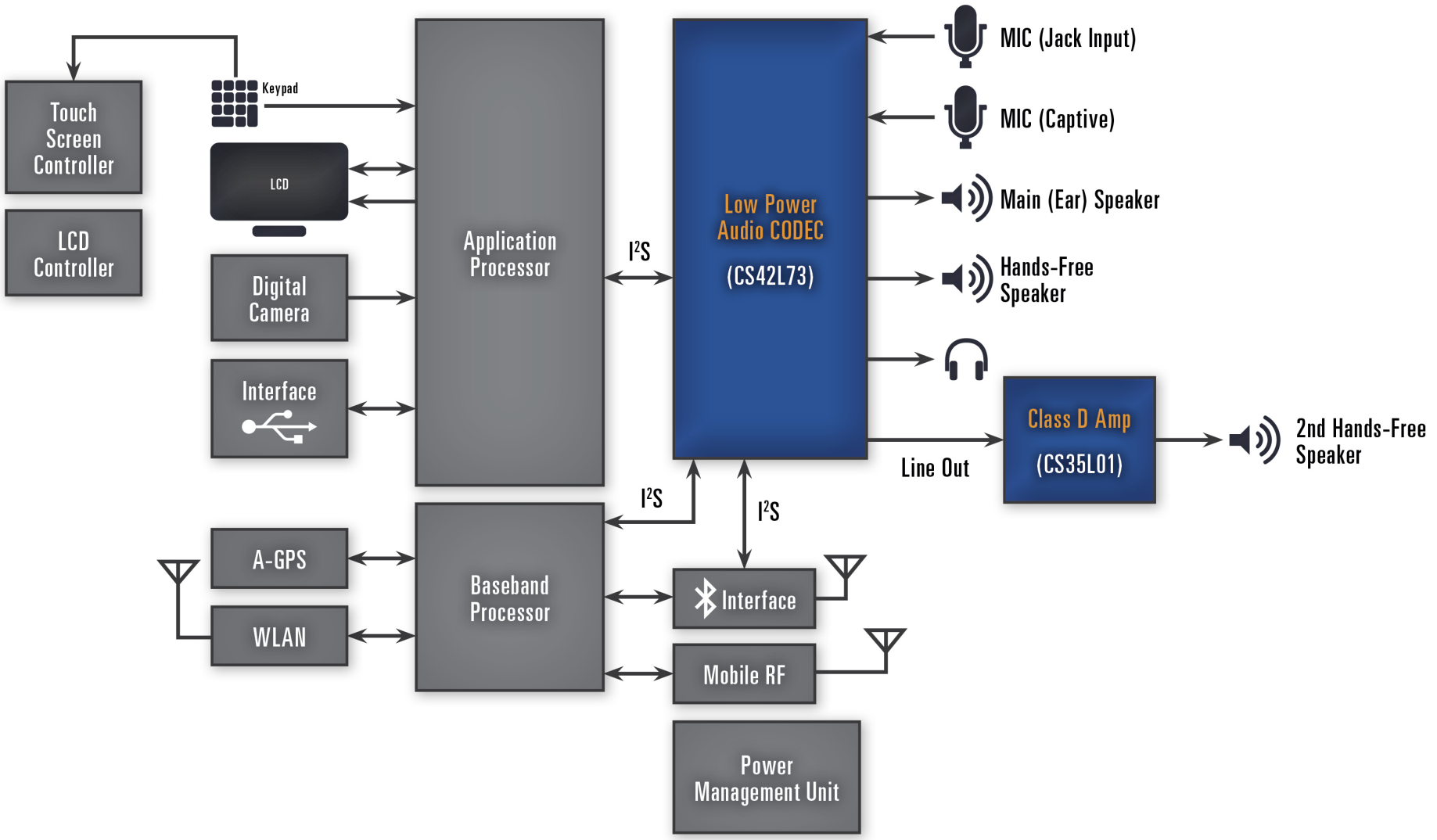
LI. Jaime Jesús Delgado Meraz
j2deme+movil@gmail.com

Instituto Tecnológico de Ciudad Valles

Enero – Junio 2013

Introducción

- Los smartphones, son los dispositivos de información más prometedores en el campo de la computación móvil.
- Con poderosas capacidades de cómputo, comunicación y sensores, incrementandose día a día, estos dispositivos han evolucionado desde simples dispositivos de comunicación hacia consumidores de diversos servicios de información.
- Programar aplicaciones que puedan ejecutarse en este nuevo entorno de cómputo no es fácil del todo, mientras que por un lado, soluciones como los middleware, ofrecen una capa de abstracción funcional, un punto importante en el diseño para moviles es la conciencia de recursos disponibles.


Touch Screen Controller
LCD Controller
Keypad
LCD
Digital Camera
Interface
Application Processor
I²S
Low Power Audio CODEC
(CS42L73)
MIC (Jack Input)
MIC (Captive)
Main (Ear) Speaker
Hands-Free Speaker
Class D Amp
(CS35L01)
2nd Hands-Free Speaker
Line Out
I²S
I²S
A-GPS
WLAN
Baseband Processor
Interface
Mobile RF
Power Management Unit

Características

- Los smartphones son plataformas relativamente poderosas de cómputo móvil
- Ofrecen capacidades razonables de procesamiento de datos y almacenamiento, incorporan varias tecnologías de comunicación, y usualmente incluyen dispositivo embebidos tales como cámaras y sensores.
- La característica más importante de los smartphones, como *habilitadores* del cómputo móvil, es la posibilidad de instalar nuevas aplicaciones.

Hardware

- **CPUs**: Generalmente se utilizan procesadores de arquitectura ARM debido a su eficiencia de energía. La frecuencia de reloj de los modelos actuales oscila entre los 200 - 300 MHz, con algunos más potentes alcanzando los GHz.
- **Memoria**: La cantidad de RAM oscila entre los 64 y 128 Mb. Por el lado del almacenamiento persistente, esta provisto mediante almacenamiento externo con memorias flash de hasta 64 GB, y memorias internas de hasta 8 GB, pero que no pueden ser usadas como extensiones de la RAM.
- **Batería**: La mayoría de los smartphones utilizan baterías de litio-ion debido a su reducido tamaño, y rangos de carga de entre 780 y 1,200 miliamperes/hora, lo cual se traduce en varios días de operación en espera o algunas horas de llamadas telefónicas.

- La principal tecnología de comunicación de datos en los smartphones dentro de la red telefónica móvil es la modalidad de intercambio de paquetes, ya sea mediante Enhanced Data Rates for GSM Evolution (EDGE) o el Universal Mobile Telecomunnication systems (UTMS).
- Algunas redes solo tienen disponible el canal de baja velocidad General Packet Radio Service (GPRS).
- Las tecnologías Bluetooth, Infrarrojo y NFC estan disponibles para comunicación de corto alcance. Actualmente la mayoría de los dispositivos calificados como smartphones también ofrecen conectividad Wi-Fi (IEEE 802.11b/g)

- Un smartphone es más que un dispositivo de comunicacíon, sino que también funciona como un media center con la inclusión de cámaras y reproductores de video.
- Los receptores de GPS (Global Positioning System) también estan disponibles en la mayoría de los modelos actuales de smartphones de manera interna, o compatible como accesorio externo.
- En lo concerniente al diseño, los tamaños de las pantallas y su resolución, el tamaño del teclado y el diseño del keypad han avanzado, facilitando el uso de las aplicaciones.

Entornos de Desarrollo

- Los lenguajes de programación, frameworks y otras herramientas de desarrollo, existen principalmente para resolver la complejidad del software mediante el uso de abstracción.
  - Ya sea usando las mismas metodologías y herramientas de aplicaciones basadas en servidores y aplicaciones de escritorio, y usandolas para desarrollar aplicaciones moviles.
  - Aunque existen muchos aspectos del diseño de software para móvil e implementaciones, que no estan contempladas en los frameworks y herramientas actuales, es muy grande lo que las aplicaciones moviles y estacionarias comparten.
  - Los frameworks y herramientas para el desarrollo de aplicaciones moviles estan evolucionando constantemente en el desarrollo de técnicas arquitecturales e innovaciones que permitan satisfacer las dimensiones de movilidad, y todos los conjuntos de características permiten definir una taxonomía.

Software Stack

- Al construir sobre un kernel (núcleo) y otras características de bajo nivel, el software utilizado en los dispositivos moviles puede caracterizarse dentro de tres diferentes categorías:
  - **Software de aplicación**: Se refiere al desarrollo de aplicaciones de utilidad para el usuario final. La implementación de componentes de software puede beneficiarse del uso de interfaces provistas para los componentes de bajo nivel, pero sus implementaciones no se basan del todo en estos.

Software Stack

- Al construir sobre un kernel (núcleo) y otras características de bajo nivel, el software utilizado en los dispositivos moviles puede caracterizarse dentro de tres diferentes categorías:
  - **Middleware**: ofrece facilidades para el desarrollo de aplicaciones. Comúnmente incluye librerías para dar soporte para el uso de ciertos protocolos de comunicación. Es común que las implementaciones deban adecuarse a alguna interfaces predefinidas. El enfasis se da en la portabilidad en el sentido de que los detalles de la capa de hardware subyacente no son siempre beneficiosos, aún cuando resulten en un desempeño mejorado.

Software Stack

- Al construir sobre un kernel (núcleo) y otras características de bajo nivel, el software utilizado en los dispositivos moviles puede caracterizarse dentro de tres diferentes categorías:
  - **Software de bajo nivel**: Cubre el kernel y los drivers del dispositivo, y la máquinas virtuales cuando así corresponda. Usualmente todo este software esta definido por el fabricante. El desarrollo de software de bajo nivel recuerda al desarrollo de sistemas profundamente embebidos, con una cercana conexión al hardware subyacente.

Desarrollo de aplicaciones

- La definición más básica de una aplicación es una pieza de software que puede ser iniciada y terminada individualmente y que realiza una tarea.
- Sin embargo, es también necesario asociar una interfaz de usuario con una aplicación, puesto que de otra manera, observar el comportamiento de la aplicación sería difícil.
- Una aplicación puede verse como una pieza de código ejecutable, que puede ser activada por el usuario o el sistema bajo alguna condición dada.
- Para poder decir que una pieza de software es una aplicación, se debe definir un mecánismo de interfaz, que indique a la infraestructura de ejecución que estamos trabajando con una aplicación.

Integración de aplicaciones

- Un efecto de como se debe usar una aplicación, es la profundidad de la integración de esta con el resto del sistema:

  Aplicación Independiente Estas aplicaciones esta contenidas a si mismas.

  Aplicación librería de código compartido En principio, esto sucede cuando la plataforma provee de servicios a nuevas aplicaciones.

  Aplicación que comparte en directo con otra Por ejemplo, se puede generar una aplicación administración de información personal y agenda, que recolecte datos de contactos, lista de tareas, calendario y otras.

  Embebidas Algunas aplicaciones pueden simplemente estar dentro de otras.

Flujo de desarrollo de aplicación

- Un flujo de trabajo común en el desarrollo de aplicaciones para el ambiente móvil, con especial enfoque en la usabilidad y actividades de usuario consiste en:
  1. Scoping (definición de alcances).
  2. Consideraciones de rendimiento.
  3. Diseño de la interfaz de usuario.
  4. Modelo de datos y problemas de memoria.
  5. Comunicación y Entrada/Salida (I/O).

Scoping

- Se debe definir el propósito fundamental de la aplicación, incluyendo que tanto lo que nuestra aplicación puede y no puede hacer.
- Se deben tomar en cuenta las características físicas del dispositivo, sobre todo si es que estas implican una restricción.

Consideraciones de rendimiento

- Las métricas generales de responsividad son necesarias para las aplicaciones. Esto incluye, por ejemplo, que tan rápido se debe abrir un menú en la aplicación.
- La responsividad total debe ser tomada en cuenta como una parte importante de la experiencia de usuario.
- Comenzar con características clave y su rendimiento, y continuar con aquellas menos importantes solo cuando las más importantes tengan un nivel aceptable de rendimiento.
- Se debe considerar que sobre-enfocar el rendimiento puede ser dañino para la portabilidad de la aplicación.

Diseño de la interfaz de usuario

- Además del scoping y las innovaciones, se debe considerar la productividad del usuario final y la responsividad como los principios más importantes del diseño de la interfaz de usuario.
- Lo anterior significa que las acciones que son típicas y naturales para el usuario final puedan ser ejecutadas rápida y fácilmente.
- Esto último, significa que el usuario tenga la sensación de control al realizar tareas, lo que comúnmente implica minimizar el tiempo que el usuario tiene que esperar para que las actividades se completen, y aún más importante, nunca se debe dejar al usuario pensando en que estará haciendo el dispositivo.

Modelo de Datos y problemas de memoria

- La manera en se representa los datos, impacta en como debe ser ubicada en la memoria, y en como el sistema se comporta en condiciones "pico" (momentos de máxima carga), así como también de como la aplicación desecha los datos.
- Las estructuras de datos y uso de memoria en general, siempre deben de considerarse cuidadosamente.

Comunicación y Entrada/Salida

- La manera en que la comunicación e I/O (entrada/salida) sean definidas, determina como la aplicación se comunica con los recursos que estan más alla de su control.
- Esto incluye los recursos internos del dispositivo, tales como archivos y subsistemas, así como también los recursos externos del dispositivo, y que requieren un mecanismo de comunicación, previo al acceso.

Programación Basada en Eventos

- Un paradigma comúnmente usado en la programación de interfaces gráficas de usuario es el permitir a cada elemento de la GUI generar eventos.
- Los "callbacks" (llamadas de retorno) son usados para enlazar eventos y código. Estos son operaciones especiales donde se puede registrar una operación que es llamada cuando determinado evento, digamos una tecla presionada, ocurre en la ejecución.
- La diferencia entre la programación basada en eventos con los paradigmas tradicionales de programación, es que en el paradigma tradicional, quien solicita el servicio sabe quien lo provee, mientras que en el modelo basado en eventos, la fuente del evento solo lo crea, y corresponde a los componentes registrados para manejar el evento, realizar sus respectivas acciones.

Modelo-Vista-Controlador

- También conocido como **MVC**, es un paradigma de arquitectura de aplicaciones, cuyo propósito es separar todas las funciones de la aplicación en una de tres categorías:

  Modelo El cual contiene las estructuras de datos de la aplicación.

  Vista Las cuales contienen las interfaces de usuario de la aplicación.

  Controlador Que permite al usuario controlar el comportamiento de la aplicación.

- **Modelo**: El rol del modelo en patrón MVC, es contener toda la información sin importar los datos de la aplicación. Además, todas las operaciones concernientes a los datos estan incluidas, siguiendo los lineamientos de encapsulación de datos. El modelo es también, usualmente responsable de la administración de los datos en la memoria permanente, puesto que puede guardar y cargar los datos a necesidad.
- **Vista**: Las responsabilidades de la vista en el patrón de diseño MVC, estan relacionadas a mostrar la información al usuario. Cada que un modelo se actualice, las vistas registradas para observar el modelo son notificadas sobre la actualización. Entonces, solicitan la información acerca de la actualización y muestran los datos actualizados. Una sola aplicación, puede tener múltiples vistas para diferentes propósitos.

- **Controlador**: Las responsabilidades del controlador dentro del modelo MVC, estan relacionadas a controlar la aplicación. La implementación más común es que un controlador escuche los comandos que la aplicación recibe y los transforme de manera que pueda ser interpretada por el modelo.

Administración de Aplicaciones

- Se puede asociar un gran número de operaciones a las aplicaciones que se ejecutan en móvil:
  - *Obtención*: Las aplicaciones pueden ser obtenidas de alguna ubicacíon en la red o por algún otro tipo de medio. Por ejemplo, medios físicos, como tarjetas SD, pueden ser usados. Sin embargo, para los dispositivos inalámbricos, descargar aplicaciones "en el aire" (OTA, *Over the Air*), es probablemente la alternativa más exitosa.
  - *Instalación*: La instalación puede incluir múltiples pasos intermedios, tales como verificar que la instalación este permitida, y la transformación, donde el software descargado puede ser transformado en una aplicación ejecutable.
  - *Lanzamiento*: Las aplicaciones deben, obviamente, poder ser ejecutadas una vez que han sido descargadas e instaladas en el dispositivo.

Administración de Aplicaciones

- Se puede asociar un gran número de operaciones a las aplicaciones que se ejecutan en móvil:
  - *Control de versiones*: En algún momento será necesario actualizar las aplicaciones instaladas, cuando una nueva versión se libere. También es posible que actualizar una versión del subsistema (o aplicación) requiera también una actualización de otras partes del sistema.
  - *Desinstalación*: Las aplicaciones pueden removerse del dispositivo, cuando ya no se necesiten, o que el espacio de almacenamiento ocupado por ellas se necesite para otro uso.